aws

Panduan Developerr

AWS Step Functions

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

AWS Step Functions: Panduan Developerr

Copyright © 2024 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 antara para pelanggan, atau dengan cara apa pun yang menghina atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan kekayaan masing-masing pemiliknya, yang mungkin berafiliasi, terkait dengan, atau disponsori oleh Amazon, atau mungkin tidak.

Table of Contents

Apa itu AWS Step Functions? 1
AWS SDK dan Integrasi yang Dioptimalkan 2
Alur kerja Standar dan Ekspres 2
Spesifikasi alur kerja standar 2
Spesifikasi alur kerja ekspres 2
Kasus penggunaan 3
Kasus penggunaan #1: Orkestrasi fungsi 3
Kasus penggunaan #2: Percabangan 4
Kasus penggunaan #3: Penanganan kesalahan 4
Kasus penggunaan #4: Manusia dalam lingkaran 5
Kasus penggunaan #5: Pemrosesan paralel 6
Kasus penggunaan #6: Paralelisme dinamis 6
Integrasi layanan 7
Wilayah yang didukung 11
Apakah ini pertama kalinya Anda menggunakan Step Functions? 11
Prasyarat 12
Langkah 1: Buat akun dan pengguna IAM 12
Daftar Akun AWS 12
Membuat pengguna administratif 13
Langkah 2: Berikan akses terprogram 14
Mulai 16
Konsep kunci 16
Tutorial dalam seri ini 18
Tutorial 1: Buat prototipe untuk mesin negara Anda 22
Langkah selanjutnya 24
Tutorial 2: Tentukan integrasi layanan pertama menggunakan fungsi Lambda 24
Langkah 1: Buat dan uji fungsi Lambda 24
Langkah 2: Perbarui alur kerja - konfigurasikan status Get credit limit 25
Langkah selanjutnya 26
Tutorial 3: Menerapkan kondisi jika-lain dalam alur kerja Anda 26
Langkah 1: Membuat topik Amazon SNS yang menerima token callback 27
Langkah 2: Buat fungsi Lambda untuk menangani callback 28
Langkah 3: Perbarui alur kerja-tambahkan logika kondisi jika-lain dalam keadaan Choice 30
Langkah selanjutnya 32

Tutorial 4: Tentukan beberapa tugas untuk dilakukan secara paralel 33
Langkah 1: Buat fungsi Lambda untuk melakukan pemeriksaan yang diperlukan 33
Langkah 2: Perbarui alur kerja - Tambahkan tugas paralel yang akan dilakukan 35
Tutorial 5: Bersamaan mengulangi koleksi item 36
Langkah 1: Buat tabel DynamoDB untuk menyimpan nama semua biro kredit 37
Langkah 2: Perbarui mesin negara - Ambil hasil dari tabel DynamoDB 38
Langkah 3: Buat fungsi Lambda yang mengembalikan skor kredit untuk semua biro kredit 38
Langkah 4: Perbarui mesin status - tambahkan status Peta untuk mengambil skor kredit secara berulang 39
Tutorial 6: Simpan alur kerja dan jalankan mesin negara 40
Langkah 1: Simpan mesin negara 40
Langkah 2: Tambahkan kebijakan IAM yang tersisa 41
Langkah 3: Jalankan mesin negara 42
Tutorial 7: Konfigurasikan input dan output 43
Pilih bagian tertentu dari input mentah menggunakan InputPath filter 44
Memanipulasi masukan yang dipilih menggunakan filter Parameter 48
Mengkonfigurasi output menggunakanResultSelector,ResultPath, dan OutputPath filter 49
Tutorial 8: Kesalahan debug di konsol 52
Debugging galat status Choice jalur tidak valid 52
Debugging kesalahan ekspresi jalur JSON saat menerapkan filter input dan output 55
Kasus penggunaan 57
Pemrosesan data 57
Machine Learning 59
Orkestrasi mikro 60
IT dan otomatisasi keamanan 61
Cara Step Functions bekerja 63
Alur Kerja Standar vs Ekspres 63
Alur kerja Ekspres Sinkron dan Tidak Sinkron 67
Jaminan eksekusi 68
Optimalisasi biaya menggunakan Alur Kerja Ekspres 69
Status 71
Amazon States Language 74
Diteruskan 95
Tugas 96
Pilihan 118
Tunggu 124

Berhasil 126
Gagal 127
Paralel 129
Map 134
Mode pemrosesan status peta 134
Mode sebaris dan perbedaan mode Terdistribusi 135
Menggunakan status Peta dalam mode Inline 137
Menggunakan status Peta dalam mode Terdistribusi 146
Ambang kegagalan yang ditoleransi untuk status Peta Terdistribusi 157
Transisi 159
Transisi dalam status Peta Terdistribusi 160
Data Mesin Status 161
Format Data 161
Input/Output Mesin Status 162
Input/output status 162
Pengolahan Input dan Output 164
Jalan 166
InputPath, Parameter dan ResultSelector 168
ResultPath 174
OutputPath 183
InputPath,ResultPath, dan OutputPath Contoh 184
Bidang input dan output negara peta 189
Objek konteks 221
Simulator aliran data 227
Menggunakan simulator aliran data 228
Pertimbangan simulator aliran data 230
Versi dan alias 231
Versi 232
Alias 236
Otorisasi untuk versi dan alias 239
Mengaitkan eksekusi dengan versi atau alias 242
Contoh penyebaran 246
Penyebaran versi secara bertahap 248
Eksekusi 258
Mulai Eksekusi dari Tugas 258
Menggunakan EventBridge Scheduler 261

Eksekusi Alur Kerja Standar dan Ekspres 268
Melihat dan men-debug eksekusi 273
Redrivingeksekusi 295
Memeriksa Map Run 306
Penanganan kesalahan 319
Nama kesalahan 319
Mencoba kembali setelah kesalahan 322
Status fallback 327
Nyatakan contoh mesin menggunakan Coba Ulang dan menggunakan Catch 329
Panggil Step Functions 334
Konsistensi Baca 335
Penandaan di Step Functions 335
Penandaan untuk Alokasi Biaya 336
Penandaan untuk Keamanan 337
Melihat dan Mengelola 337
Menandai API 338
Workflow Studio 339
Gambaran umum antarmuka 340
Mode desain 341
Mode kode 347
Modus Config 351
Pintasan keyboard 355
Menggunakan Workflow Studio 355
Buat alur kerja 356
Rancang alur kerja 358
Jalankan alur kerja Anda 365
Edit alur kerja 366
Ekspor alur kerja Anda 368
Buat prototipe alur kerja Anda 369
Konfigurasikan input dan output 370
Konfigurasikan input ke status 371
Konfigurasikan output status 374
Peran eksekusi di Workflow Studio 380
Tentang peran yang dibuat secara otomatis 381
Secara otomatis menghasilkan peran 381
Menyelesaikan masalah pembuatan peran 383

Peran untuk menguji Tugas HTTP di Workflow Studio 384
Peran untuk menguji integrasi layanan yang dioptimalkan di Workflow Studio 384
Peran untuk menguji integrasi layanan AWS SDK di Workflow Studio 385
Peran untuk menguji status alur di Workflow Studio 385
Penanganan kesalahan 386
Coba lagi pada kesalahan 387
Tangkap kesalahan 388
Timeout 388
HeartbeatSeconds 388
Tutorial: Belajar untuk menggunakan Studio Alur Kerja AWS Step Functions 389
Langkah 1: Arahkan ke Workflow Studio 390
Langkah 2: Buat mesin negara 390
Langkah 3: Tinjau definisi Bahasa Amazon States yang dibuat secara otomatis 392
Langkah 4: Edit definisi alur kerja dalam mode Kode 394
Langkah 5: Simpan mesin negara 396
Langkah 6: Jalankan mesin negara 397
Langkah 7: Perbarui mesin negara Anda 398
Langkah 8: Membersihkan 399
Tutorial 401
Buat mesin status Step Functions yang menggunakan Lambda 401
Langkah 1: Membuat fungsi Lambda 402
Langkah 2: Uji fungsi Lambda 403
Langkah 3: Buat mesin negara 404
Langkah 4: Jalankan mesin negara 406
Penanganan Kondisi Kesalahan Menggunakan Mesin Status 408
Langkah 1: Buat fungsi Lambda yang gagal 408
Langkah 2: Uji fungsi Lambda 409
Langkah 3: Buat mesin status dengan bidang Catch 410
Langkah 4: Jalankan mesin negara 412
Ulangi tindakan menggunakan status Peta Inline 414
Langkah 1: Buat prototipe alur kerja 414
Langkah 2: Konfigurasikan input dan output 415
Langkah 3: Tinjau definisi Bahasa Amazon States yang dibuat secara otomatis dan simpan alur kerjanya 416
Langkah 4: Jalankan mesin negara 418
Memulai dengan menggunakan status Peta Terdistribusi 419

Prasyarat 420
Langkah 1: Buat prototipe alur kerja 420
Langkah 2: Konfigurasikan bidang yang diperlukan untuk status Peta 421
Langkah 3: Konfigurasikan opsi tambahan 422
Langkah 4: Konfigurasikan fungsi Lambda 423
Langkah 5: Perbarui prototipe alur kerja 424
Langkah 6: Tinjau definisi Bahasa Amazon States yang dibuat secara otomatis dan simpan alur kerjanya 424
Langkah 7: Jalankan mesin negara 427
Memproses seluruh batch data dengan fungsi Lambda 428
Langkah 1: Buat mesin negara 428
Langkah 2: Buat fungsi Lambda 430
Langkah 3: Jalankan mesin negara 431
Memproses item data individual dengan fungsi Lambda 433
Langkah 1: Buat mesin negara 434
Langkah 2: Buat fungsi Lambda 436
Langkah 3: Jalankan mesin negara 431
Memulai Eksekusi Mesin Status dalam Respons terhadap Peristiwa Amazon S3 441
Prasyarat: Buat Mesin Status 441
Langkah 1: Buat Bucket di Amazon S3 442
Langkah 2: Aktifkan Pemberitahuan Acara Amazon S3 dengan EventBridge 442
Langkah 3: Buat EventBridge Aturan Amazon 443
Langkah 4: Uji Aturan 444
Contoh Input Eksekusi 445
Membuat Step Functions API menggunakan API Gateway 445
Langkah 1: Buat IAM role untuk API Gateway 446
Langkah 2: Buat API Gateway API Anda 447
Langkah 3: Uji dan Deploy API Gateway API 450
Membuat mesin kondisi Step FunctionsAWS SAM 452
Prasyarat 453
Langkah 1: Unduh Sampel Aplikasi AWS SAM 454
Langkah 2: Bangun Aplikasi Anda 455
Langkah 3: Deploy Aplikasi Anda ke Cloud AWS 456
Pemecahan Masalah 457
Bersihkan 458
Membuat mesin status Aktivitas 458

Langkah 1: Buat Aktivitas 459
Langkah 2: Buat mesin negara 460
Langkah 3: Terapkan Pekerja 461
Langkah 4: Jalankan mesin negara 464
Langkah 5: Jalankan dan Hentikan Pekerja 465
Ulangi loop dengan Lambda 466
Langkah 1: Buat fungsi Lambda untuk mengulangi hitungan 466
Langkah 2: Uji Fungsi Lambda 467
Langkah 3: Buat Mesin Status 468
Langkah 4: Mulai Eksekusi Baru 471
Melanjutkan Pekerjaan Berkelanjutan sebagai Eksekusi Baru 472
Menggunakan aksi Step Functions API (disarankan) 473
Menggunakan fungsi Lambda 477
Men-deploy Contoh Proyek Persetujuan Manusia 489
Langkah 1: Buat Templat 490
Langkah 2: Buat tumpukan 490
Langkah 3: Menyetujui langganan SNS 491
Langkah 4: Jalankan mesin negara 492
Kode Sumber Templat 494
Lihat jejak X-Ray di Step Functions 504
Langkah 1: Buat peran IAM untuk Lambda 505
Langkah 2: Buat fungsi Lambda 505
Langkah 3: Buat dua fungsi Lambda lagi 507
Langkah 4: Buat mesin negara 507
Langkah 5: Jalankan mesin negara 510
Kumpulkan info bucket Amazon S3 menggunakan integrasi layanan AWS SDK 513
Langkah 1: Buat mesin negara 513
Langkah 2: Tambahkan izin peran IAM yang diperlukan 516
Langkah 3: Jalankan eksekusi mesin status Standar 516
Langkah 4: Jalankan eksekusi mesin status Express 517
Alat developer 519
Opsi Pengembangan 519
Konsol Step Functions 520
AWS SDK 520
Alur kerja Standar dan Ekspres 521
API Layanan HTTPS 521

Lingkungan pengembangan ... 521
Titik akhir ... 522
AWS CLI ... 522
Step Functions Local ... 523
AWS Toolkit for Visual Studio Code ... 523
AWS Serverless Application Model dan Step Functions ... 523
Terraform dan Step Functions ... 524
Dukungan format ketetapan ... 524
Step Functions dan AWS SAM ... 531
Mengapa menggunakan Step Functions dengan AWS SAM? ... 532
Integrasi Step Functions dengan spesifikasi AWS SAM ... 532
Integrasi Step Functions dengan SAM CLI ... 532
DefinitionSubstitutions dalam AWS SAM template ... 534
Langkah selanjutnya ... 537
Menggunakan Workflow Studio di Application Composer ... 538
Menggunakan Workflow Studio di Application Composer ... 539
Referensi sumber daya secara dinamis menggunakan substitusi CloudFormation definisi ... 539
Connect tugas integrasi layanan ke kartu komponen yang disempurnakan ... 540
Impor proyek yang ada dan sinkronkan secara lokal ... 541
Fitur Workflow Studio yang tidak tersedia di Komposer Aplikasi AWS ... 541
Membuat Mesin Negara Lambda Menggunakan AWS CloudFormation ... 542
Langkah 1: Siapkan AWS CloudFormation template Anda ... 542
Langkah 2: Gunakan AWS CloudFormation template untuk membuat Lambda State Machine ... 548
Langkah 3: Mulai eksekusi Mesin Status ... 553
Membuat mesin Lambda negara menggunakan AWS CDK ... 554
Langkah 1: Siapkan proyek AWS CDK Anda ... 555
Langkah 2: Gunakan AWS CDK untuk membuat mesin negara ... 556
Langkah 3: Mulai eksekusi mesin negara ... 565
Langkah 4: Bersihkan ... 566
Langkah selanjutnya ... 566
Membuat API REST API Gateway dengan Mesin Status Ekspres Sinkron Menggunakan AWS CDK ... 567
Langkah 1: Siapkan Proyek AWS CDK Anda ... 568
Langkah 2: Gunakan AWS CDK untuk membuat API Gateway REST API dengan integrasi backend Synchronous Express State Machine ... 571

Langkah 3: Uji API Gateway 581
Langkah 4: Bersihkan 584
SDK Ilmu Data 584
Menyebarkan mesin status menggunakan Terraform 585
Prasyarat 585
Siklus hidup pengembangan dengan Terraform 586
Peran dan kebijakan IAM untuk mesin negara Anda 588
Pengujian dan Debugging 590
Menggunakan TestState API 590
Pertimbangan tentang penggunaan API TestState 591
Menggunakan level inspeksi di TestState API 592
IAMizin untuk menggunakan API TestState 599
Menguji status (Konsol) 600
Menguji status menggunakan AWS CLI 601
Menguji dan men-debug aliran data input dan output 607
Menguji mesin negara secara lokal 611
Menyiapkan Step Functions Lokal (Versi yang Dapat Diunduh) dan Docker 612
Menyiapkan Step Functions Lokal (Versi yang Dapat Diunduh) - Versi Java 613
Mengatur Opsi Konfigurasi untuk Step Functions Lokal 614
Menjalankan Step Functions Lokal di Komputer Anda 616
Pengujian Fungsi Langkah dan AWS SAM CLI lokal 618
Menggunakan Integrasi Layanan Mocked 623
Praktik terbaik 642
Gunakan timeout untuk menghindari eksekusi macet 642
Gunakan ARN Amazon S3 bukan meneruskan muatan besar 643
Hindari mencapai kuota sejarah 645
Menangani pengecualian layanan Lambda 646
Hindari latensi saat polling untuk tugas aktivitas 647
Memilih Alur Kerja Standar atau Ekspres 648
Pembatasan ukuran kebijakan sumber daya Amazon CloudWatch Logs 649
Bekerja dengan layanan yang lain 650
Hubungi AWS layanan lain 650
Integrasi yang dioptimalkan 651
AWS Integrasi SDK 651
Dukungan pola integrasi 651
Akses lintas akun 655

AWS Integrasi layanan SDK 655
Menggunakan AWS integrasi layanan SDK 656
Layanan yang didukung 657
Tindakan API yang tidak didukung untuk layanan yang didukung 699
Integrasi layanan SDK yang tidak digunakan AWS lagi 701
Integrasi yang dioptimalkan 701
Amazon API Gateway 705
Amazon Athena 713
AWS Batch 715
Amazon Bedrock 717
AWS CodeBuild 721
Amazon DynamoDB 727
Amazon ECS/Fargate 730
Amazon EKS 733
Amazon EMR 748
Amazon EMR di EKS 760
Amazon EMR Serverless 764
Amazon EventBridge 773
AWS Glue 775
AWS Glue DataBrew 776
AWS Lambda 777
AWS Elemental MediaConvert 781
Amazon SageMaker 784
Amazon SNS 794
Amazon SQS 797
AWS Step Functions 800
Panggil API pihak ketiga 804
Definisi Tugas HTTP 804
Bidang Tugas HTTP 805
Otentikasi untuk Tugas HTTP 812
Menggabungkan EventBridge koneksi dan data definisi Tugas HTTP 813
Menerapkan pengkodean URL pada badan permintaan 816
Izin IAM untuk menjalankan Tugas HTTP 817
Contoh Tugas HTTP 819
Menguji Tugas HTTP 821
Respons HTTP Task yang tidak didukung 823

Pola integrasi layanan 824
Minta Tanggapan 824
Jalankan Tugas (.sync) 825
Tunggu Panggilan Balik dengan Token Tugas 827
Meneruskan parameter ke API layanan 833
Lulus JSON statis sebagai parameter 833
Lulus masukan status sebagai parameter menggunakan Jalur 834
Meneruskan Simpul Objek Konteks sebagai Parameter 834
Ubah log untuk integrasi 835
Proyek sampel untuk Step Functions 859
Mengelola pekerjaan batch (AWS Batch,Amazon SNS) 860
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan 860
Langkah 2: Jalankan mesin negara 862
Contoh Kode Mesin Status 864
Contoh IAM 865
Mengelola tugas kontainer (Amazon ECS,Amazon SNS) 866
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan 866
Langkah 2: Jalankan mesin negara 868
Contoh Kode Mesin Status 869
Contoh IAM 871
Mentransfer catatan data (Lambda,DynamoDB,Amazon SQS) 872
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan 873
Langkah 2: Jalankan mesin negara 875
Contoh Kode Mesin Status 876
Contoh IAM 878
Polling untuk Status Pekerjaan (Lambda,) AWS Batch 879
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan 880
Langkah 2: Jalankan mesin negara 882
Contoh Kode Mesin Status 884
Timer Tugas (Lambda, Amazon SNS) 886
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan 887
Langkah 2: Jalankan mesin negara 889
Contoh Pola Panggilan Balik (Amazon SQS, Amazon SNS, Lambda) 891
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan 892
Langkah 2: Jalankan mesin negara 894
Contoh Panggilan Balik Lambda 896

Kelola Tugas Amazon EMR 896
Langkah 1: Buat Mesin Negara dan Sumber Daya Penyediaan 897
Langkah 2: Jalankan mesin negara 868
Contoh Kode Mesin Status 869
Contoh IAM 871
Jalankan EMR Serverless pekerjaan 905
Templat AWS CloudFormation dan sumber daya tambahan 906
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan 906
Langkah 2: Jalankan mesin negara 908
Mulai Alur Kerja dalam Alur Kerja (Step Functions, Lambda) 909
Langkah 1: Buat mesin negara dan sumber daya penyediaan 909
Langkah 2: Jalankan mesin negara 912
Contoh Kode Mesin Status 913
Secara dinamis memproses data dengan status Peta 915
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan 915
Langkah 2: Berlangganan topik Amazon SNS 919
Langkah 3: Tambahkan pesan ke antrian Amazon SQS 919
Langkah 4: Jalankan mesin negara 920
Contoh kode mesin status 921
Contoh IAM 923
Memproses file CSV dengan Peta Terdistribusi 924
AWS CloudFormation template dan sumber daya tambahan 925
Langkah 1: Buat mesin negara dan sumber daya penyediaan 926
Langkah 2: Jalankan mesin negara 928
Memproses data dalam bucket Amazon S3 dengan Peta Terdistribusi 930
AWS CloudFormation template dan sumber daya tambahan 931
Langkah 1: Buat mesin negara dan sumber daya penyediaan 932
Langkah 2: Jalankan mesin negara 935
Melatih Model Machine Learning 936
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan 936
Langkah 2: Jalankan mesin negara 938
Contoh Kode Mesin Status 940
Contoh IAM 942
Setel Model Machine Learning 943
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan 944
Langkah 2: Jalankan mesin negara 946

Contoh Kode Mesin Status 948
Contoh IAM 952
Proses Pesan Bervolume Tinggi dari Amazon SQS (Alur Kerja Express) 955
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan 956
Langkah 2: Memicu eksekusi mesin negara 958
Contoh Kode Fungsi Lambda 959
Contoh Kode Mesin Status 960
Contoh IAM 961
Contoh Checkpointing selektif (Alur kerja Express) 962
Langkah 1: Buat Mesin Negara dan Sumber Daya Penyediaan 963
Langkah 2: Jalankan mesin negara 965
Contoh Kode Mesin Status untuk Induk (Standar Alur Kerja) 967
Contoh IAM role untuk Mesin Status Induk 969
Contoh Kode Mesin Status untuk Mesin Status Nest (Alur Kerja Express) 967
Contoh IAM role untuk Mesin Status Anak 973
Membangun AWS CodeBuild Proyek (CodeBuild, Amazon SNS) 974
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan 975
Langkah 2: Jalankan mesin negara 977
Contoh Kode Mesin Status 978
Praproses data dan latih model machine learning 980
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan 981
Langkah 2: Jalankan mesin negara 983
Contoh Kode Mesin Status 984
Contoh IAM 988
Contoh orkestrasi Lambda 989
Langkah 1: Buat mesin negara dan sumber daya penyediaan 990
Langkah 2: Jalankan mesin negara 992
Tentang mesin negara dan pelaksanaannya 994
Contoh IAM 997
Mulai kueri Athena 999
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan 1000
Langkah 2: Jalankan mesin negara 1002
Contoh Kode Mesin Status 1003
Contoh IAM 1005
Jalankan beberapa kueri (Amazon Athena, Amazon SNS) 1007
Langkah 1: Buat mesin negara dan sumber daya penyediaan 1007

Langkah 2: Jalankan mesin negara 1011
Contoh Kode Mesin Status 1011
Contoh IAM 1014
Kueri kumpulan data besar (Amazon Athena, Amazon S3,, AWS Glue Amazon SNS) 1018
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan 1018
Langkah 2: Jalankan mesin negara 1021
Contoh Kode Mesin Status 1022
Contoh IAM 1024
Tetap perbarui data (Amazon Athena, Amazon S3,) AWS Glue 1027
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan 1027
Langkah 2: Jalankan mesin negara 1029
Contoh Kode Mesin Status 1030
Contoh IAM 1031
Mengelola klaster Amazon EKS 1034
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan 1034
Langkah 2: Jalankan mesin negara 1037
Contoh Kode Mesin Status 1038
Contoh IAM 1042
Buat panggilan ke API Gateway 1044
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan 1044
Langkah 2: Jalankan mesin negara 1046
Contoh Kode Mesin Status 1047
Contoh IAM 1049
Panggil layanan mikro dengan API Gateway 1050
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan 1050
Langkah 2: Jalankan mesin negara 1053
Contoh Kode Mesin Status 1054
Contoh IAM 1055
Kirim acara khusus ke EventBridge 1057
Langkah 1: Buat mesin negara dan sumber daya penyediaan 1057
Langkah 2: Jalankan mesin negara 1059
Contoh Kode Mesin Status 1060
Contoh IAM 1061
Memanggil Alur Kerja Express Sinkron 1062
Langkah 1: Buat mesin negara dan sumber daya penyediaan 1062
Langkah 2: Jalankan mesin negara 1065

Contoh Kode Mesin Status 1066
Contoh IAM 1068
Jalankan alur kerja ETL/ELT menggunakan Amazon Redshift 1069
Langkah 1: Buat mesin negara dan sumber daya penyediaan 1070
Langkah 2: Jalankan mesin negara 1072
Contoh Kode Mesin Status 1074
Contoh IAM 1094
Gunakan Step Functions dan AWS Batch dengan penanganan kesalahan 1095
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan 1095
Langkah 2: Jalankan mesin negara 1097
Contoh Kode Mesin Status 1098
Contoh IAM 1100
Menggemari AWS Batch pekerjaan 1101
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan 1101
Langkah 2: Jalankan mesin negara 1104
Contoh Kode Mesin Status 1105
Contoh IAM 1106
AWS Batch dengan Lambda 1107
Langkah 1: Buat Mesin Negara dan Sumber Daya Penyediaan 1107
Langkah 2: Jalankan mesin negara 1109
Contoh Kode Mesin Status 1111
Contoh IAM 1112
Lakukan AI prompt-chaining dengan Amazon Bedrock 1113
Templat AWS CloudFormation dan sumber daya tambahan 1114
Prasyarat 1114
Langkah 1: Buat mesin negara dan sumber daya penyediaan 1114
Langkah 2: Jalankan mesin negara 1117
Kuota 1119
Kuota umum 1120
Kuota yang terkait dengan akun 1121
Kuota yang terkait dengan HTTP Task 1122
Kuota terkait throttling status 1122
Kuota terkait throttling tindakan API 1123
Kuota yang terkait dengan API TestState 1124
Kuota lainnya 1124
Kuota yang berkaitan dengan eksekusi mesin status 1128

Kuota yang berkaitan dengan eksekusi tugas 1131
Kuota yang terkait dengan versi dan alias 1132
Pembatasan terkait penandaan 1132
Pencatatan dan pemantauan 1134
CloudWatch Metrik Amazon 1134
Metrik yang melaporkan interval waktu 1135
Metrik yang melaporkan hitungan 1136
Metrik eksekusi 1136
Metrik jumlah sumber daya untuk versi dan alias 1139
Metrik Aktivitas 1140
Metrik Fungsi Lambda 1141
Metrik Integrasi Layanan 1142
Metrik Layanan 1143
Metrik API 1144
Pengiriman CloudWatch metrik upaya terbaik 1145
Melihat Metrik untuk Step Functions 1145
Pengaturan Alarm untuk Step Functions 1147
EventBridge Acara Amazon 1149
EventBridge muatan 1150
Contoh peristiwa Step Functions 1151
Merutekan acara Step Functions ke EventBridge 1155
Mencatat Step Functions Menggunakan AWS CloudTrail 1157
Langkah Fungsi Informasi di CloudTrail 1157
Contoh: Entri Berkas Log Step Functions 1158
Logging menggunakanCloudWatchLog 1163
Konfigurasikan log 1164
CloudWatchLog muatan 1164
Kebijakan IAM untuk log keCloudWatchLog 1165
Tingkat Log 1166
X-Ray 1170
Penyiapan dan konfigurasi 1172
Konsep 1176
Integrasi layanan 1177
Melihat konsol X-Ray 1178
Melihat informasi pelacakan X-Ray untuk Step Functions 1178
Pelacakan 1178

Peta layanan ... 1179
Segmen dan subsegmen ... 1180
Analitik ... 1182
Konfigurasi ... 1183
Bagaimana jika tidak ada data dalam peta jejak atau peta layanan? ... 1184
Menggunakan Notifikasi Pengguna AWS dengan Step Functions ... 1185
Keamanan ... 1186
Perlindungan Data ... 1186
Enkripsi ... 1187
Manajemen Identitas dan Akses ... 1187
Audiens ... 1188
Mengautentikasi dengan identitas ... 1188
Mengelola akses menggunakan kebijakan ... 1192
Kontrol Akses ... 1195
Tindakan kebijakan ... 1195
Sumber daya kebijakan ... 1196
Kunci persyaratan kebijakan ... 1197
ACL ... 1198
ABAC ... 1198
Kredensial sementara ... 1199
Izin pengguna utama ... 1199
Peran layanan ... 1200
Peran terkait layanan ... 1200
Bagaimana AWS Step Functions bekerja dengan IAM ... 1201
Contoh kebijakan berbasis identitas ... 1202
Kebijakan berbasis identitas ... 1205
Kebijakan berbasis sumber daya ... 1205
AWS kebijakan terkelola ... 1206
Membuat peran IAM mesin negara ... 1208
Membuat Izin IAM Terperinci untuk Pengguna Non-Admin ... 1211
Mengakses sumber daya lintas akun AWS ... 1214
VPC Endpoint ... 1225
Kebijakan IAM untuk layanan terintegrasi ... 1228
Kebijakan IAM untuk menggunakan status Peta Terdistribusi ... 1319
Kebijakan berbasis tanda ... 1325
Memecahkan masalah ... 1326

Pencatatan dan Pemantauan 1328
Validasi Kepatuhan 1328
Ketangguhan 1329
Keamanan Infrastruktur 1330
Analisis Konfigurasi dan Kelemahan 1330
Migrasi beban kerja dari AWS Data Pipeline 1331
Memigrasi beban kerja 1331
Pemetaan konsep 1332
Proyek contoh Step Functions 1333
Perbandingan harga 1334
Pemecahan Masalah 1335
Pemecahan masalah umum 1335
Saya tidak bisa membuat mesin status. 1335
Saya tidak dapat menggunakan JsonPath untuk referensi output tugas sebelumnya. 1335
Ada penundaan dalam transisi status. 1336
Ketika saya mulai eksekusi alur kerja standar baru, mereka gagal dengan kesalahan `ExecutionLimitExceeded`. 1336
Kegagalan pada satu cabang dalam keadaan paralel menyebabkan seluruh eksekusi gagal. 1336
Memecahkan masalah integrasi layanan 1337
Pekerjaan saya selesai di layanan hilir, tetapi dalam Step Functions status tugas tetap “Sedang berlangsung” atau penyelesaiannya tertunda. 1337
Saya ingin mengembalikan output JSON dari eksekusi mesin status nest. 1337
Saya tidak bisa memanggil fungsi Lambda dari akun lain. 1337
Saya tidak dapat melihat token tugas diteruskan dari status `.waitForTaskToken`. 1339
Memecahkan masalah aktivitas 1339
Eksekusi mesin negara saya terjebak pada keadaan aktivitas. 1339
Waktu pekerja aktivitas saya habis saat menunggu token tugas. 1340
Memecahkan masalah alur kerja ekspres 1340
Aplikasi saya habis waktunya sebelum menerima tanggapan dari StartSyncExecution Panggilan API. 1340
Saya tidak dapat melihat sejarah eksekusi untuk memecahkan masalah kegagalan alur kerja ekspres. 1340
Informasi terkait 1342
Peluncuran fitur terbaru 1343
Riwayat dokumen 1346

AWSGlosarium 1390
..... mcccxci

Apa itu AWS Step Functions?

AWS Step Functions adalah layanan orkestrasi tanpa server yang memungkinkan Anda berintegrasi dengan AWS Lambdafungsi dan lainnya Layanan AWS untuk membangun aplikasi bisnis yang penting. Melalui konsol grafis Step Functions, Anda melihat alur kerja aplikasi Anda sebagai serangkaian langkah yang digerakkan oleh peristiwa.

Step Functions didasarkan pada mesin status dan tugas. Dalam Step Functions, alur kerja disebut state machine, yang merupakan serangkaian langkah yang digerakkan oleh peristiwa. Setiap langkah dalam alur kerja disebut status. TugasNegara mewakili unit kerja yang dilakukan oleh AWS layanan lain, seperti AWS Lambda,. Status Tugas dapat memanggil API Layanan AWS atau apa pun.

Dengan kontrol bawaan Step Functions, Anda memeriksa status setiap langkah dalam alur kerja Anda guna memastikan bahwa aplikasi Anda berjalan sesuai urutan dan seperti yang diharapkan. Bergantung pada kasus penggunaan Anda, Anda dapat memiliki AWS layanan panggilan Step Functions, seperti Lambda, untuk melakukan tugas. Anda dapat membuat alur kerja yang memproses dan memublikasikan model machine learning. Anda dapat memiliki AWS layanan kontrol Step Functions, seperti AWS Glue, untuk membuat alur kerja ekstrak, transformasi, dan beban (ETL). Anda juga dapat membuat alur kerja otomatis yang berjalan lama untuk aplikasi yang memerlukan interaksi manusia.

> **Tip**
>
> Untuk membiasakan diri dengan fitur utama Step Functions melalui serangkaian modul interaktif, lihat The AWS Step Functions Workshop. Atau mulailah menggunakan Step Functions dengan mengikuti tutorial Memulai ini untuk membuat alur kerja aplikasi kartu kredit.

Topik

- AWS SDK dan Integrasi yang Dioptimalkan
- Alur kerja Standar dan Ekspres
- Kasus penggunaan
- Integrasi layanan
- Wilayah yang didukung
- Apakah ini pertama kalinya Anda menggunakan Step Functions?

AWS SDK dan Integrasi yang Dioptimalkan

Untuk memanggil AWS layanan lain, Anda dapat menggunakan integrasi AWS SDK Fungsi Langkah, atau Anda dapat menggunakan salah satu integrasi yang Dioptimalkan oleh Fungsi Langkah.

- Integrasi AWS SDK memungkinkan Anda memanggil salah satu dari lebih dari dua ratus AWS layanan langsung dari mesin status Anda, memberi Anda akses ke lebih dari sembilan ribu tindakan API.
- Integrasi yang dioptimalkan oleh Step Functions telah disesuaikan untuk menyederhanakan penggunaan di mesin status Anda.

Alur kerja Standar dan Ekspres

Step Functions memiliki dua tipe alur kerja. Alur kerja standar memiliki eksekusi alur kerja tepat satu kali dan dapat berjalan hingga satu tahun. Ini berarti bahwa setiap langkah dalam alur kerja Standar akan dijalankan tepat sekali. Alur kerja ekspres, bagaimanapun, memiliki eksekusi at-least-once alur kerja dan dapat berjalan hingga lima menit. Ini berarti bahwa satu atau beberapa langkah dalam Alur Kerja Ekspres berpotensi berjalan lebih dari satu kali, sementara setiap langkah dalam alur kerja dijalankan setidaknya sekali.

Eksekusi adalah instans tempat Anda menjalankan alur kerja untuk melakukan tugas. Alur kerja standar adalah ideal untuk alur kerja yang berjalan lama dan dapat diaudit, karena menunjukkan riwayat eksekusi dan debugging visual. Alur kerja ekspres ideal untuk high-event-rate beban kerja, seperti pemrosesan data streaming dan konsumsi data IoT.

Spesifikasi alur kerja standar

- Tingkat eksekusi 2.000 per detik
- Tingkat transisi status 4,000 per detik
- Harga berdasarkan transisi negara
- Tampilkan riwayat eksekusi dan debugging visual
- Mendukung semua integrasi dan pola layanan

Spesifikasi alur kerja ekspres

- Tingkat eksekusi 100.000 per detik

- Tingkat transisi status yang hampir tidak terbatas
- Harga berdasarkan jumlah dan durasi eksekusi
- Kirim riwayat eksekusi ke Amazon CloudWatch
- Tampilkan riwayat eksekusi dan debugging visual berdasarkan level Log yang diaktifkan
- Mendukung semua integrasi layanan dan sebagian besar pola

Untuk informasi selengkapnya tentang alur kerja Standar dan Ekspres, termasuk harga Step Functions, lihat berikut ini:

- Alur Kerja Standar vs Ekspres
- AWS Step Functions harga

Kasus penggunaan

Step Functions mengelola komponen dan logika aplikasi Anda, sehingga Anda dapat menulis lebih sedikit kode dan fokus untuk membangun dan memperbarui aplikasi Anda dengan cepat. Bagian ini menjelaskan kasus penggunaan umum untuk bekerja dengan Step Functions.

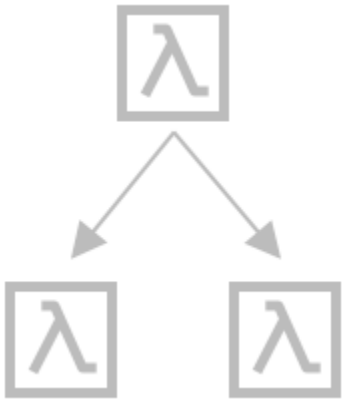
Kasus penggunaan #1: Orkestrasi fungsi

Anda membuat alur kerja yang menjalankan sekelompok fungsi Lambda (langkah-langkah) dalam urutan tertentu. Satu output fungsi Lambda diteruskan ke input fungsi Lambda berikutnya. Langkah terakhir dalam alur kerja Anda memberikan hasil. Dengan Step Functions, Anda dapat melihat setiap langkah dalam alur kerja Anda berinteraksi satu sama lain, sehingga Anda dapat memastikan bahwa setiap langkah tersebut menjalankan fungsi yang diinginkan.

Untuk tutorial yang menunjukkan cara membuat mesin status dengan sekelompok fungsi, lihat berikut ini:

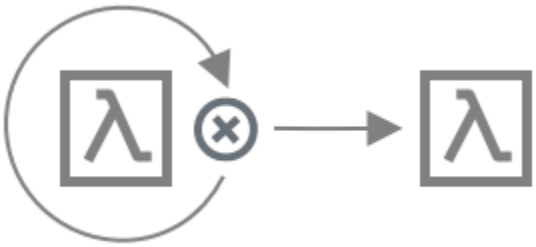
- Memulai dengan AWS Step Functions

Kasus penggunaan #2: Percabangan



Pelanggan meminta peningkatan batas kredit. Dengan menggunakan status `Choice`, Anda dapat meminta Step Functions membuat keputusan berdasarkan input status `Choice`. Jika permintaan lebih dari batas kredit pelanggan Anda yang telah disetujui sebelumnya, Anda dapat meminta Step Functions mengirimkan permintaan pelanggan Anda ke pengelola untuk sign-off. Jika permintaan kurang dari batas kredit pelanggan Anda yang telah disetujui sebelumnya, Anda dapat meminta Step Functions untuk menyetujui permintaan tersebut secara otomatis.

Kasus penggunaan #3: Penanganan kesalahan



`Retry`

Dalam kasus penggunaan ini, pelanggan meminta nama pengguna. Pertama kali, permintaan pelanggan Anda tidak berhasil. Dengan menggunakan pernyataan `Retry`, Anda dapat meminta Step Functions untuk mencoba permintaan pelanggan Anda kembali. Kedua kalinya, permintaan pelanggan Anda berhasil.

`Catch`

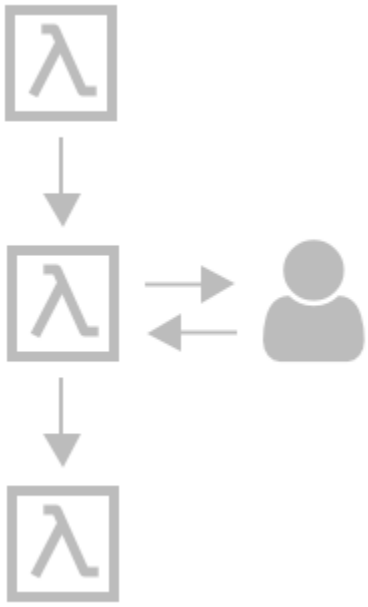
Dalam kasus penggunaan yang sama, pelanggan meminta nama pengguna yang tidak tersedia. Dengan menggunakan pernyataan `Catch`, Anda meminta Step Functions menyarankan nama pengguna yang tersedia. Jika pelanggan Anda mengambil nama pengguna yang tersedia, Anda

dapat meminta Step Functions untuk menuju ke langkah berikutnya dalam alur kerja Anda, yaitu mengirim email konfirmasi. Jika pelanggan Anda tidak mengambil nama pengguna yang tersedia, Anda meminta Step Functions menuju ke langkah yang berbeda dalam alur kerja Anda, yaitu memulai proses pendaftaran dari awal.

Untuk contoh pernyataan `Retry` dan `Catch` yang lebih mendetail, lihat berikut ini:

- Penanganan kesalahan dalam Step Functions

Kasus penggunaan #4: Manusia dalam lingkaran

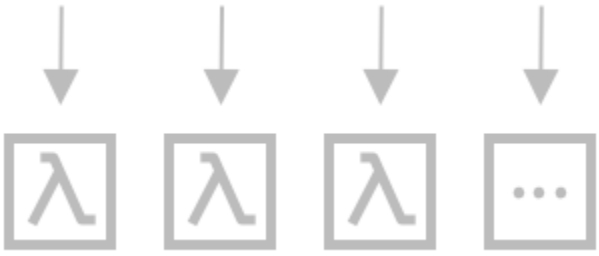


Dengan menggunakan aplikasi perbankan, salah satu pelanggan Anda mengirim uang ke seorang teman. Pelanggan Anda menunggu email konfirmasi. Dengan panggilan balik dan token tugas, Anda meminta Step Functions memberi tahu Lambda untuk mengirim uang pelanggan Anda dan melaporkan kembali saat teman pelanggan Anda menerimanya. Setelah Lambda melaporkan kembali bahwa teman pelanggan Anda menerima uang, Anda dapat meminta Step Functions menuju ke langkah berikutnya dalam alur kerja Anda, yaitu mengirim email konfirmasi kepada pelanggan Anda.

Untuk melihat contoh proyek yang memperlihatkan panggilan balik dengan token tugas, lihat hal berikut:

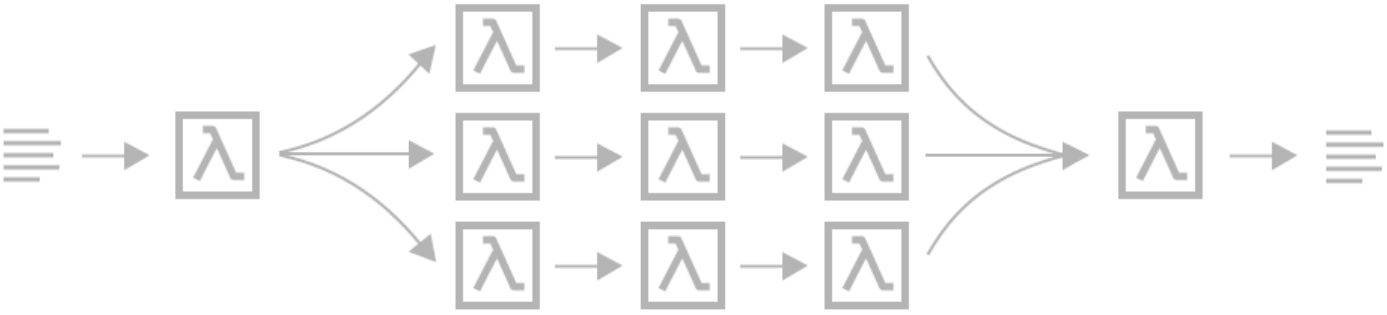
- Contoh Pola Panggilan Balik (Amazon SQS, Amazon SNS, Lambda)

Kasus penggunaan #5: Pemrosesan paralel



Pelanggan mengonversi file video menjadi lima resolusi tampilan yang berbeda, sehingga pemirsa dapat menonton video di beberapa perangkat. Dengan menggunakan status `Parallel`, Step Functions memasukkan file video, sehingga Lambda dapat memprosesnya menjadi lima resolusi tampilan secara bersamaan.

Kasus penggunaan #6: Paralelisme dinamis



Seorang pelanggan memesan tiga item, dan Anda perlu menyiapkan setiap item untuk pengiriman. Anda memeriksa ketersediaan setiap item, mengumpulkan setiap item, lalu mengemas setiap item untuk pengiriman. Dengan menggunakan status `Map`, Step Functions membuat Lambda memproses setiap item pelanggan Anda secara paralel. Setelah semua item pelanggan Anda dikemas untuk pengiriman, Step Functions melanjutkan ke langkah berikutnya dalam alur kerja Anda, yaitu mengirimkan email konfirmasi kepada pelanggan Anda dengan informasi pelacakan.

Untuk melihat contoh proyek yang menunjukkan paralelisme dinamis menggunakan status `Map`, lihat hal berikut:

- Secara dinamis memproses data dengan status Peta

Integrasi layanan

Step Functions terintegrasi dengan beberapa AWS layanan. Untuk menggabungkan Step Functions dengan layanan ini, gunakan pola integrasi layanan berikut:

Minta tanggapan (default)

- Panggil layanan, dan biarkan Step Functions berprogres ke status berikutnya setelah mendapat respons HTTP.

Jalankan pekerjaan (.sync)

- Panggil layanan, dan minta Step Functions menunggu tugas untuk diselesaikan.

Tunggu panggilan balik dengan token tugas (. waitForTaskToken)

- Panggil layanan dengan token tugas, dan minta Step Functions menunggu hingga token tugas kembali dengan panggilan balik.

Tabel di bawah ini menunjukkan integrasi layanan dan pola integrasi layanan yang tersedia untuk Step Functions.

Alur Kerja Standar dan Alur Kerja Ekspres mendukung integrasi yang sama tetapi bukan pola integrasi yang sama.

- Dukungan pola integrasi yang dioptimalkan berbeda untuk setiap integrasi.
- Alur Kerja Ekspres tidak mendukung Run a Job (.sync) atau Wait for Callback (. waitForTaskToken).
- Untuk informasi selengkapnya, lihat Alur Kerja Standar vs Ekspres.

Standard Workflows

Integrasi layanan yang didukung

	Layanan	Permintaan Respon	Jalankan Job (`.sync`)	Tunggu Callback () `.waitForTaskToken`
Integrasi yang dioptimalkan	Amazon API Gateway	✓		✓
	Amazon Athena	✓	✓	
	AWS Batch	✓	✓	
	Amazon Bedrock	✓	✓	✓
	AWS CodeBuild	✓	✓	
	Amazon DynamoDB	✓		
	Amazon ECS/Fargate	✓	✓	✓
	Amazon EKS	✓	✓	✓
	Amazon EMR	✓	✓	
	Amazon EMR on EKS	✓	✓	
	Amazon EMR Serverless	✓	✓	
	Amazon EventBridge	✓		✓
	AWS Glue	✓	✓	
	AWS Glue DataBrew	✓	✓	
	AWS Lambda	✓		✓
	AWS Elemental MediaConvert	✓	✓	

	Layanan	Permintaan Respon	Jalankan Job (`.sync`)	Tunggu Callback () `.waitForTaskToken`
	Amazon SageMaker	✓	✓	
	Amazon SNS	✓		✓
	Amazon SQS	✓		✓
	AWS Step Functions	✓	✓	✓
AWS Integrasi SDK	Lebih dari dua ratus	✓		✓

Express Workflows

Integrasi layanan yang didukung

	Layanan	Permintaan Respon	Jalankan Job (`.sync`)	Tunggu Callback () `.waitForTaskToken`
Integrasi yang dioptimalkan	Amazon API Gateway	✓		
	Amazon Athena	✓		
	AWS Batch	✓		
	Amazon Bedrock	✓		
	AWS CodeBuild	✓		
	Amazon DynamoDB	✓		

	Layanan	Permintaan Respon	Jalankan Job (`.sync`)	Tunggu Callback () `.waitForTaskToken`
	Amazon ECS/Fargate	✓		
	Amazon EKS	✓		
	Amazon EMR	✓		
	Amazon EMR on EKS	✓		
	Amazon EMR Serverless	✓		
	Amazon EventBridge	✓		
	AWS Glue	✓		
	AWS Glue DataBrew	✓		
	AWS Lambda	✓		
	AWS Elemental MediaConvert	✓		
	Amazon SageMaker	✓		
	Amazon SNS	✓		
	Amazon SQS	✓		
	AWS Step Functions	✓		
AWS Integrasi SDK	Lebih dari dua ratus	✓		

Wilayah yang didukung

Sebagian besar AWS wilayah mendukung Step Functions. Untuk daftar lengkap AWS wilayah tempat Step Functions tersedia, lihat Tabel AWS Wilayah.

Apakah ini pertama kalinya Anda menggunakan Step Functions?

Jika ini adalah pertama kalinya Anda menggunakan Step Functions, topik berikut membantu Anda memahami berbagai bagian kerja dengan Step Functions, termasuk bagaimana Step Functions digabungkan dengan AWS layanan lain:

- Tutorial untuk Step Functions
- Proyek sampel untuk Step Functions
- AWS Step Functions SDK Ilmu Data untuk Python

Prasyarat untuk Memulai AWS Step Functions

Sebelum Anda mulai dengan AWS Step Functions untuk pertama kalinya, lengkapi prasyarat yang tercantum di halaman ini.

Topik

- Langkah 1: Mendaftar untuk Akun AWS dan pengguna IAM
- Langkah 2: Berikan akses terprogram

Langkah 1: Mendaftar untuk Akun AWS dan pengguna IAM

Untuk mengakses layanan AWS, Anda harus membuat Akun AWS terlebih dahulu. Anda dapat menggunakan Akun AWS untuk melihat laporan aktivitas dan penggunaan Anda serta untuk mengelola autentikasi dan akses. Anda hanya dikenakan biaya untuk produk dan layanan yang Anda gunakan, dan Anda dapat memulai dengan AWS gratis. Untuk informasi selengkapnya, lihat AWS Tingkat Gratis.

Untuk menghindari penggunaan tindakan Step Functions AndaPengguna root akun AWS, ini adalah praktik terbaik untuk membuat pengguna IAM untuk setiap orang yang membutuhkan akses administratif ke Step Functions.

Jika Anda sudah memiliki akun AWS, lewati ke prasyarat berikutnya.

Daftar Akun AWS

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

Untuk mendaftar Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.
2. Ikuti petunjuk secara online.

   Anda akan diminta untuk menerima panggilan telepon dan memasukkan kode verifikasi pada keypad telepon sebagai bagian dari prosedur pendaftaran.

   Saat Anda mendaftar Akun AWS, Pengguna root akun AWS dibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya dalam akun. Sebagai praktik terbaik

keamanan, tetapkan akses administratif ke pengguna administratif, dan hanya gunakan pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

AWS akan mengirimkan email konfirmasi kepada Anda setelah proses pendaftaran selesai. Anda dapat melihat aktivitas akun saat ini dan mengelola akun dengan mengunjungi https://aws.amazon.com/ dan memilih Akun Saya.

Membuat pengguna administratif

Setelah Anda mendaftarAkun AWS, amankanPengguna root akun AWS, aktifkanAWS IAM Identity Center, dan buat pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Mengamankan Pengguna root akun AWS Anda

1. Masuk ke AWS Management Console sebagai pemilik akun dengan memilih Pengguna root dan memasukkan alamat email Akun AWS Anda. Pada halaman berikutnya, masukkan kata sandi Anda.

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

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

Membuat pengguna administratif

1. Aktifkan Pusat Identitas IAM.

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

   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 administratif

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

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

Langkah 2: Berikan akses terprogram

Pengguna membutuhkan akses terprogram jika mereka ingin berinteraksi dengan AWS luar dari AWS Management Console. Cara memberikan akses terprogram bergantung pada jenis pengguna yang mengakses AWS.

Untuk memberikan pengguna akses terprogram, pilih salah satu opsi berikut.

Pengguna mana yang membutuhkan akses terprogram?	Ke	Oleh
Identitas tenaga kerja (Pengguna yang dikelola di Pusat Identitas IAM)	Gunakan kredensial sementara untuk menandatangani permintaan terprogram ke AWS CLI, SDK AWS, atau API AWS.	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk AWS CLI, lihat Mengonfigurasi AWS CLI untuk menggunakan AWS IAM Identity Center di Panduan Pengguna AWS Command Line Interface. • Untuk SDK AWS, alat, dan API AWS, lihat Autentikasi Pusat Identitas IAM di Panduan Referensi SDK dan Alat AWS.
IAM	Gunakan kredensial sementara untuk menandata	Ikuti petunjuk dalam Menggunakan kredensial

Pengguna mana yang membutuhkan akses terprogram?	Ke	Oleh
	ngani permintaan terprogram ke AWS CLI, SDK AWS, atau API AWS.	sementara dengan sumber daya AWS di Panduan Pengguna IAM.
IAM	(Tidak disarankan) Gunakan kredensial jangka panjang untuk menandata ngani permintaan terprogram ke AWS CLI, AWS SDK, atau API AWS.	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk AWS CLI, lihat Mengautentikasi menggunakan kredensial pengguna IAM di Panduan Pengguna AWS Command Line Interface. • Untuk SDK dan alat AWS, lihat Mengautentikasi menggunakan kredensial jangka panjang di Panduan Referensi SDK dan Alat AWS. • Untuk API AWS, lihat Mengelola kunci akses untuk pengguna IAM di Panduan Pengguna IAM.

Memulai dengan AWS Step Functions

Selamat datang di Step FunctionsMemulaiseri tutorial.

Step Functions adalah layanan orkestrasi nirserver yang memungkinkan Anda menentukan alur kerja aplikasi sebagai serangkaian langkah yang digerakkan oleh peristiwa. Setiap langkah yang digerakkan oleh alur kerja disebut sebagainegara. Anda paling sering menggunakan negara, seperti Tugas, Pilihan, Paralel, dan Map, untuk menentukan alur kerja Anda. Di dalam `Task` menyatakan, Anda dapat menggunakanAWSIntegrasi SDK yang Step Functions mendukung dan mengatur beberapaLayanan AWSdalam alur kerja Anda.

Topik

- Konsep kunci
- Tutorial dalam seri ini
- Tutorial 1: Buat prototipe untuk mesin negara Anda
- Tutorial 2: Tentukan integrasi layanan pertama menggunakan fungsi Lambda
- Tutorial 3: Menerapkan kondisi jika-lain dalam alur kerja Anda
- Tutorial 4: Tentukan beberapa tugas untuk dilakukan secara paralel
- Tutorial 5: Bersamaan mengulangi koleksi item
- Tutorial 6: Simpan alur kerja dan jalankan mesin negara
- Tutorial 7: Konfigurasikan input dan output
- Tutorial 8: Kesalahan debug di konsol

Konsep kunci

Bagian ini memperkenalkan Anda pada konsep-konsep Step Functions yang penting. Sebelum Anda memulai, tinjau konsep kunci berikut.

Jangka waktu	Deskripsi
Alur kerja	Menjelaskan urutan langkah dan sering cocok dengan proses bisnis.

Jangka waktu	Deskripsi
Workflow Studio	Desainer alur kerja visual yang membantu Anda membuat prototipe dan membangun alur kerja lebih cepat. Untuk informasi selengkapnya, lihat AWS Step Functions Studio Alur Kerja.
Status	Langkah-langkah individual di mesin negara Anda, yang melakukan berbagai fungsi di mesin negara. Untuk informasi selengkapnya, lihat Status.
Mesin negara	Alur kerja yang didefinisikan menggunakan teks JSON yang mewakili status individu atau langkah-langkah dalam alur kerja bersama dengan bidang, sepertiStartAt,TimeoutSeconds , danVersion. Untuk informasi selengkap nya, lihat Struktur mesin status.
Amazon States Language	Sebuah bahasa yang digerakkan oleh JSON yang digunakan untuk mendefini sikan mesin negara Anda. Ini adalah koleksinegarayang dapat melakukan pekerjaan (Tasknegara), tentukan status mana yang akan dialihkan ke berikutnya (Choicenegara), dan menghentikan eksekusi dengan kesalahan (Failnegara). Untuk informasi selengkapnya, lihat Amazon States Language.
Konfigura si input dan output	Setiap negara bagian dalam alur kerja menerima data JSON sebagai input dan biasanya meneruskan data JSON sebagai output ke status berikutnya. Step Functions menyediakan beberapa filter untuk mengontrol aliran data input dan output antar status. Untuk informasi selengkapnya, lihat Pengolahan Input dan Output di Step Functions.
Integrasi layanan	Step Functions langsung terintegrasi denganLayanan AWS, memungkinkan Anda memanggil tindakan API setiap layanan dari alur kerja Anda. Untuk informasi selengkapnya, lihat Menggunakan AWS Step Functions dengan layanan lain.

Jangka waktu	Deskripsi
Jenis integrasi layanan	Step Functions menyediakan jenis integrasi layanan berikut: • Integrasi yang dioptimalkan- Disesuaikan dengan Step Functions untuk menyediakan fungsionalitas khusus untuk alur kerja. Sebagai contoh,Lambda Memohonakan mengonversi output API-nya dari string JSON yang lolos ke objek JSON. • AWSIntegrasi SDK— Berperilaku persis seperti panggilan API standar menggunakanAWSSDK. Anda dapat memanggil salah satu dari lebih dari dua ratusLayanan AWSlangsung dari mesin negara Anda dan akses lebih dari sembilan ribu tindakan API. Untuk informasi selengkapnya, lihat Menggunakan AWS Step Functions dengan layanan lain.
Pola integrasi layanan	Untuk memanggil terintegrasiLayanan AWSdalam alur kerja, Anda menggunakan salah satu pola integrasi layanan berikut yang disediakan Step Functions: • Minta respons (default)- Panggil layanan dan biarkan Step Functions maju ke status berikutnya setelah mendapat respons HTTP. • Jalankan tugas (.sync)— Hubungi layanan dan minta Step Functions menunggu pekerjaan selesai. • Tunggu panggilan balik dengan token tugas (.waitForTaskToken)— Panggil layanan dengan token tugas dan minta Step Functions menunggu hingga token tugas kembali dengan panggilan balik.
Eksekusi	Eksekusi mesin status adalah instans tempat Anda menjalankan alur kerja untuk melakukan tugas. Untuk informasi selengkapnya, lihat Eksekusi di Step Functions .

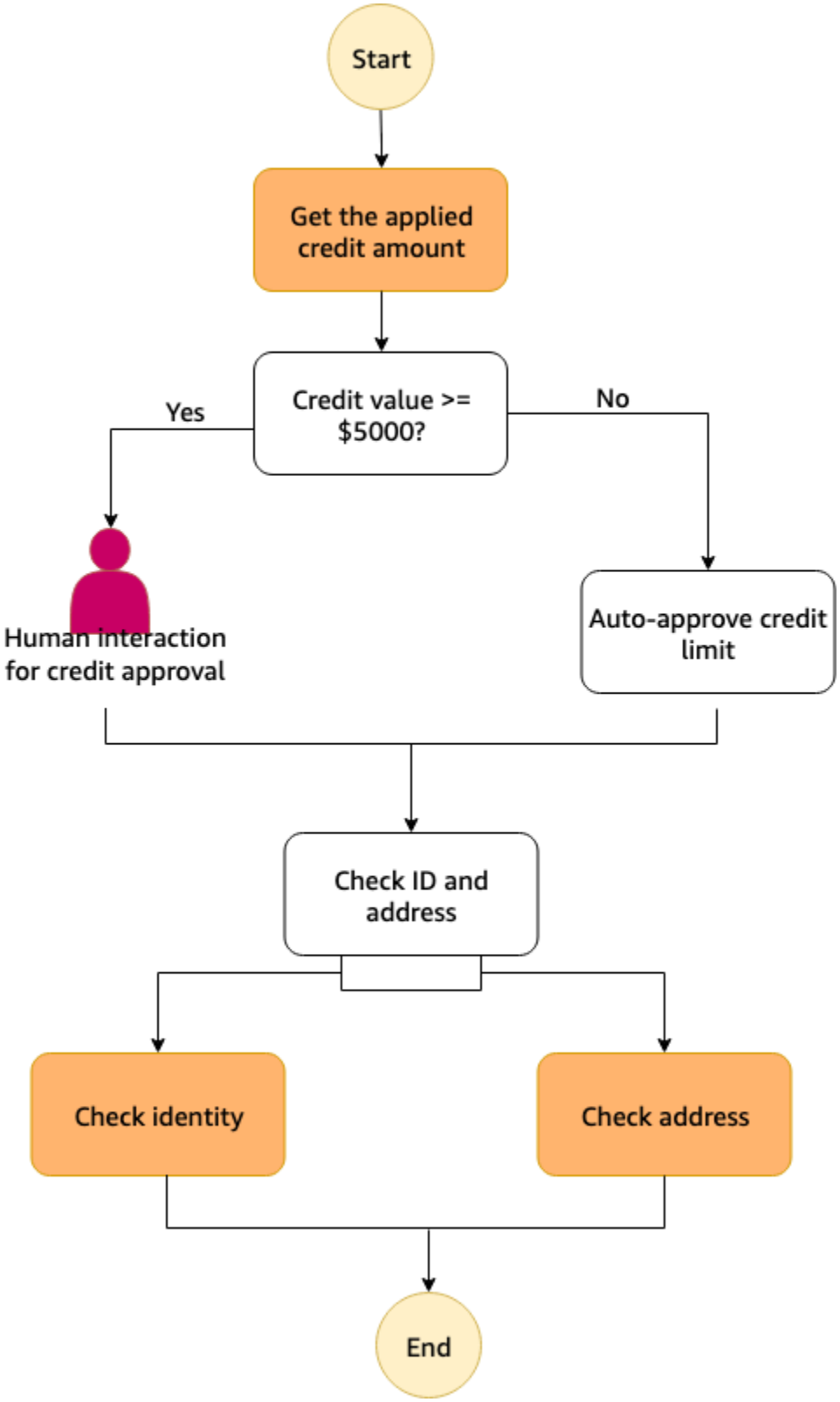
Tutorial dalam seri ini

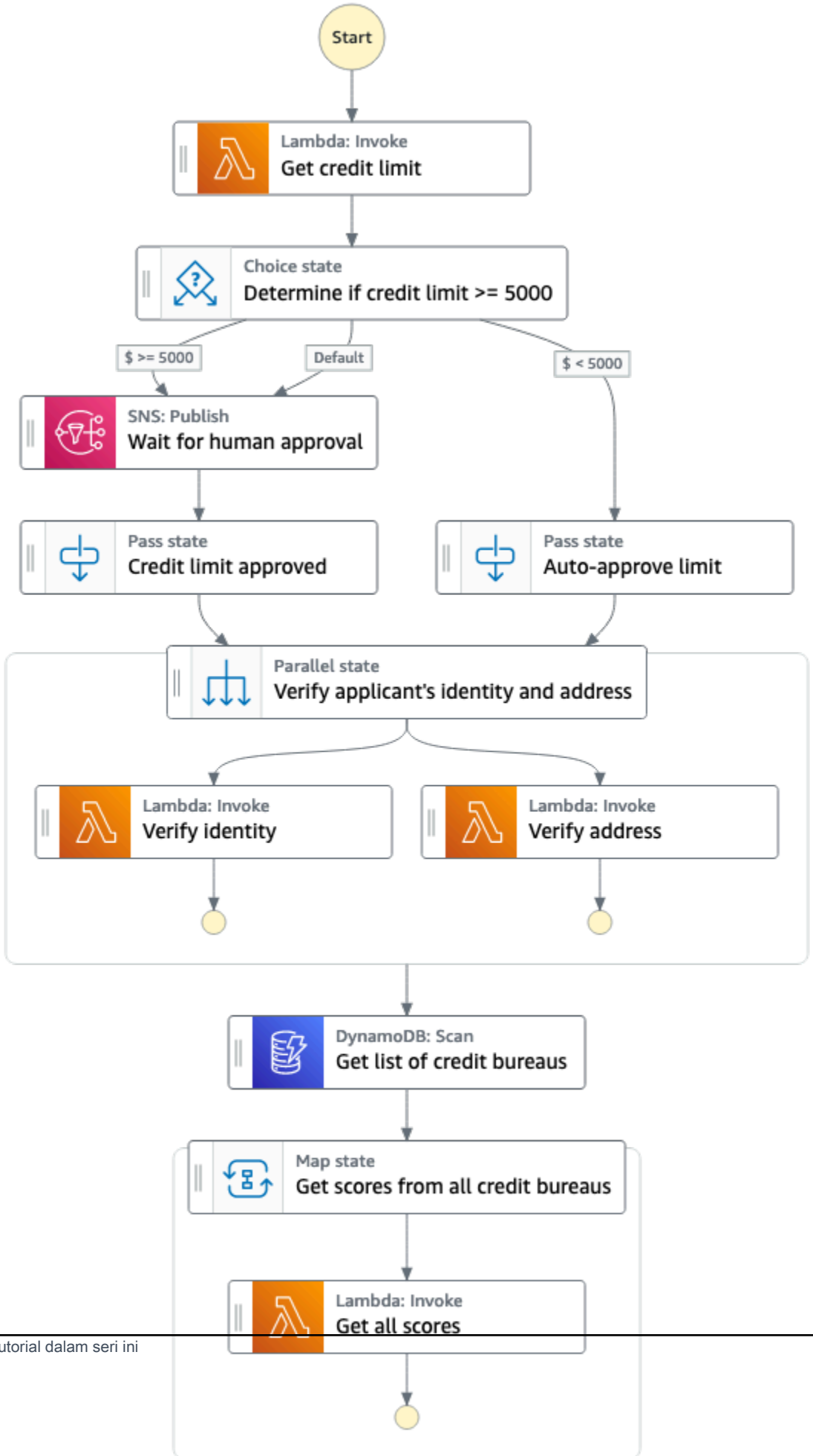
TheMemulaitutorial di bagian ini memandu Anda melalui pembuatan alur kerja dasar untuk memproses aplikasi kartu kredit. Dalam tutorial ini, Anda akan belajar cara menggunakan status yang umum digunakan di Step Functions. Anda akan mengintegrasikan alur kerja Anda dengan

yang lainLayanan AWS, sepertiAWS Lambdadan layanan pemberitahuan sederhana Amazon. Setelah menyelesaikan tutorial ini, Anda akan memiliki alur kerja sederhana yang mensimulasikan pemrosesan aplikasi kartu kredit.

> **Note**
>
> Sementara iniMemulaitutorial menggambarkan alur kerja aplikasi kartu kredit, Anda dapat menggunakan Step Functions untuk membuat beberapa jenis alur kerja. Misalnya, Anda dapat membuat alur kerja untuk pemrosesan data, otomatisasi TI, pembelajaran mesin, pemrosesan media, atau pemrosesan pesanan.

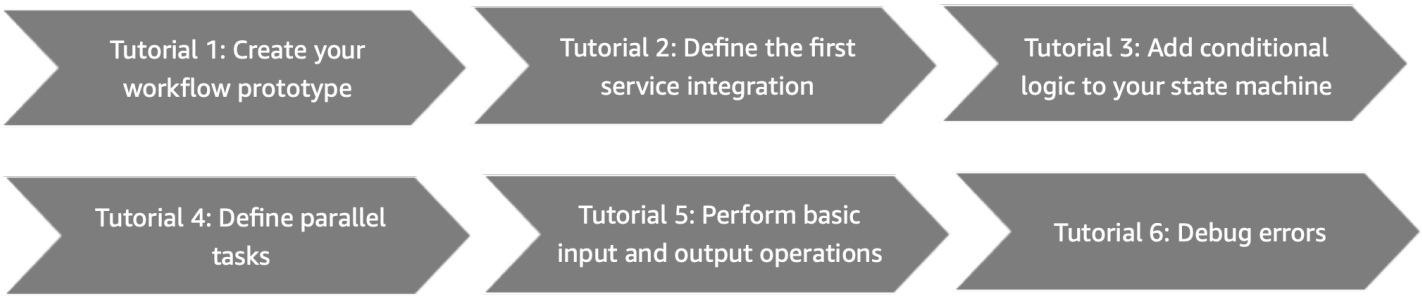
Gambar berikut mewakili alur kerja aplikasi kartu kredit dan bagaimana tampilannya saat diatur menggunakan Step Functions. Setiap langkah dalam diagram alur direpresentasikan dengan status dalam alur kerja Step Functions.


Start
Get the applied credit amount
Credit value >= $5000?
Yes
No
Human interaction for credit approval
Auto-approve credit limit
Check ID and address
Check identity
Check address
End


Start
Lambda: Invoke
Get credit limit
Choice state
Determine if credit limit >= 5000
$ >= 5000
Default
$ < 5000
SNS: Publish
Wait for human approval
Pass state
Credit limit approved
Pass state
Auto-approve limit
Parallel state
Verify applicant's identity and address
Lambda: Invoke
Verify identity
Lambda: Invoke
Verify address
DynamoDB: Scan
Get list of credit bureaus
Map state
Get scores from all credit bureaus
Lambda: Invoke
Get all scores

> **Note**
>
> Kami merekomendasikan untuk menyelesaikan semua tutorial ini secara berurutan. Menyelesaikan tutorial lengkap akan mengajarkan Anda cara menggunakan konsep dan fitur kunci untuk menggunakan Step Functions dalam alur kerja produksi.

Peta jalan berikut menunjukkan langkah-langkah yang akan Anda lakukan untuk membangun alur kerja pemrosesan kartu kredit menggunakan Studio Alur Kerja Fungsi Langkah. Langkah-langkah ini disajikan sebagai serangkaian tutorial yang berisi instruksi tentang cara menyelesaikan langkah itu.



Sebelum Anda memulai, pastikan untuk menyelesaikanprasyarat.

Tutorial 1: Buat prototipe untuk mesin negara Anda

Dalam tutorial ini, Anda membuat prototipe untuk alur kerja pemrosesan kartu kredit Anda menggunakanStudio Alur Kerja Fungsi Lambda. Anda akan memilih tindakan dan status API yang diperlukan dariTindakandanAlirantab masing-masing, dan gunakan fitur seret dan lepas Workflow Studio untuk membuat prototipe alur kerja. Dalam tutorial berikutnya, Anda akan belajar cara mengkonfigurasiLayanan AWSdan Status Fungsi Langkah yang akan Anda gunakan dalam alur kerja ini.

Untuk membuat mesin status

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. DiTentukan templatekotak dialog, pilihKosong.
3. Pilih Pilih. Ini membuka Workflow Studio diMode desain.
4. Di Workflow Studio, dariTindakantab, seretAWS LambdaMemohonTindakan API dan jatuhkan ke status kosong berlabelSeret status pertama di sini. Tentukan sebagai berikut:

    - DiKonfigurasitab, untukNama negara, masukkan**`Get credit limit`**.

5. DariAlirantab, seret dan lepasPilihannegara bagian di bawahTentukan limit kreditstatus. Ganti namaPilihanstatus**`Credit applied >= 5000?`**.
6. Seret dan jatuhkan status berikut sebagai cabang dariKredit diterapkan >= 5000?status.
    a. Tentukan Publikasikan Amazon SNS- DariTindakantab, seret dan lepasTentukan Publikasikan Amazon SNSTindakan API. Ubah nama status ini menjadi**`Wait for human approval`**.
    b. LulusNegara — DariAlirantab, seret dan lepasLulusstatus. Ubah nama cabang ini menjadi**`Auto-approve limit`**.
7. Tentukan dan jatuhkanLulusnegara bagian di bawahTunggu persetujuan manusiastatus. Tentukan nama iniLulusstatus**`Credit limit approved`**.
8. Tentukan dan jatuhkanParalelnegara setelahPilihanmenyatakan sebagai berikut:
    a. JatuhkanParalelnegara setelahBatas kredit disetujuistatus.
    b. Ganti namaParalelstatus**`Verify applicant's identity and address`**.
    c. Di bawah kedua cabangParalelnegara, seret dan lepas duaAWS LambdaMemohonTindakan API.
    d. Ubah nama negara bagian ini sebagai**`Verify identity`**dan**`Verify address`**masing-masing.
    e. PilihBatas persetujuan otomatisstatus dan untukNegara berikutnya, pilihVerifikasi identitas dan alamat pemohon.
9. TentukanPemindaian DynamoDBmenyatakan dan menjatuhkannya di bawahVerifikasi identitas dan alamat pemohonstatus. Ganti namaPemindaian DynamoDBstatus**`Get list of credit bureaus`**.
10. Tentukan dan jatuhkanPetanegara setelahDapatkan daftar biro kreditstatus. KonfigurasikanPetamenyatakan sebagai berikut:
    a. Tentukan nama menjadi**`Get scores from all credit bureaus`**.
    b. UntukModus pemrosesan, pertahankan pilihan defaultSebaris.
    c. Tentukan dan jatuhkanAWS LambdaMemohonTindakan API ke status kosong berlabelJatuhkan status di sini.
    d. Ganti namaAWS LambdaMemohonstatus**`Get all scores`**.
11. Biarkan jendela ini terbuka dan lanjutkan ke tutorial berikutnya untuk tindakan lebih lanjut.

Langkah selanjutnya

Dalam tutorial berikutnya, Anda belajar cara mengintegrasikan fungsi Lambda yang digunakan olehTentukan limit kreditstatus.

Tutorial 2: Tentukan integrasi layanan pertama menggunakan fungsi Lambda

Dalam tutorial ini, Anda belajar bagaimana mendefinisikan integrasi layanan pertama untuk alur kerja Anda. Anda menggunakan `Task` status bernama Dapatkan batas kredit untuk menjalankan fungsi Lambda. Dalam `Task` status, Anda dapat menggunakan integrasi AWS SDK yang didukung Step Functions.

Untuk menentukan integrasi layanan pertama untuk alur kerja Anda, pertama buat fungsi Lambda. Kemudian, perbarui alur kerja Anda untuk menentukan integrasi layanan dengan fungsi Lambda. Fungsi Lambda yang digunakan dalam tutorial ini mengembalikan bilangan bulat yang dihasilkan secara acak yang mewakili batas kredit yang telah diajukan pemohon.

Topik

- Langkah 1: Buat dan uji fungsi Lambda
- Langkah 2: Perbarui alur kerja - konfigurasikan status Get credit limit
- Langkah selanjutnya

Langkah 1: Buat dan uji fungsi Lambda

Anda dapat menulis kode untuk fungsi di AWS Management Console atau editor favorit Anda. Pada langkah-langkah berikut, Anda membuat fungsi Lambda Node.js berjudul. `RandomNumberforCredit`

> **Important**
>
> Pastikan bahwa prototipe alur kerja yang Anda buat di Tutorial 1 Wilayah AWS sama dengan fungsi Lambda yang akan Anda buat dalam tutorial ini.

1. Di tab atau jendela baru, buka konsol Lambda dan buat fungsi Node.js 16.x Lambda berjudul. **`RandomNumberforCredit`** Untuk informasi tentang membuat fungsi Lambda menggunakan konsol, lihat Membuat fungsi Lambda di konsol di Panduan Pengembang.AWS Lambda
2. Pada RandomNumberforCredithalaman, pilih index.mjs dan ganti kode yang ada di area sumber Kode dengan kode berikut.

```
export const handler = async function(event, context) {

    const credLimit = Math.floor(Math.random() * 10000);
    return (credLimit);

};
```

3. Dari bagian Ikhtisar fungsi, salin Nama Sumber Daya Amazon dari fungsi Lambda dan simpan dalam file teks. Anda akan memerlukan fungsi ARN sambil menentukan integrasi layanan untuk status Get credit limit. Berikut ini adalah contoh ARN:

```
arn:aws:lambda:us-east-2:123456789012:function:HelloWorld
```

4. Pilih Deploy dan kemudian pilih Uji untuk menyebarkan perubahan dan melihat output dari fungsi Lambda.

Langkah 2: Perbarui alur kerja - konfigurasikan status Get credit limit

Di konsol Step Functions, Anda akan memperbarui alur kerja untuk menentukan integrasi layanan dengan fungsi `RandomNumberforCredit` Lambda yang Anda buat di Langkah 1.

1. Buka jendela konsol Step Functions yang berisi prototipe alur kerja yang Anda buat di Tutorial 1.
2. Pilih status Get credit limit, dan di tab Configuration, lakukan hal berikut:

   a. Untuk jenis Integrasi, pertahankan pilihan default Optimized.

      Dengan menggunakan Step Functions, Anda dapat berintegrasi dengan yang lain Layanan AWS dan mengaturnya dalam alur kerja Anda. Untuk informasi selengkapnya tentang integrasi layanan dan jenisnya, lihatMenggunakan AWS Step Functions dengan layanan lain.

   b. Untuk nama Fungsi, pilih fungsi RandomNumberforCreditLambda dari daftar dropdown.

   c. Simpan pilihan default untuk item lainnya.

3. Biarkan jendela ini terbuka dan lanjutkan ke tutorial berikutnya untuk tindakan lebih lanjut.

> **Note**
>
> Dalam tutorial ini, Anda belajar bagaimana mengintegrasikan dengan fungsi Lambda dalam `Task` keadaan dalam alur kerja Anda. Anda juga dapat menggunakan integrasi AWS SDK lain yang didukung dalam `Task` status dengan menentukan nama layanan dan panggilan API, seperti yang ditunjukkan dalam sintaks berikut:
>
> ```
> arn:aws:states:::aws-sdk:serviceName:apiAction
> ```
>
> Untuk informasi selengkapnya, lihat Menggunakan AWS Step Functions dengan layanan lain.

Langkah selanjutnya

Dalam tutorial berikutnya, Anda akan menerapkan logika kondisional dalam alur kerja Anda. Logika kondisional dalam mesin status Step Functions berperilaku mirip dengan pernyataan if-else dalam bahasa pemrograman yang paling umum. Anda akan menggunakan logika kondisional dalam alur kerja Anda untuk menentukan jalur eksekusi berdasarkan informasi bersyarat.

Tutorial 3: Menerapkan kondisi jika-lain dalam alur kerja Anda

Anda dapat menerapkan kondisi if-else dalam alur kerja Anda dengan menggunakan state. `Pilihan` Ini menentukan jalur eksekusi alur kerja berdasarkan apakah kondisi tertentu mengevaluasi ke benar atau salah.

Dalam tutorial ini, Anda akan menambahkan logika kondisional untuk menentukan apakah jumlah kredit yang diterapkan dikembalikan oleh fungsi `RandomNumberforCredit` Lambda yang digunakan dalam Tutorial 2 melebihi batas ambang batas tertentu. Jika jumlahnya melebihi batas ambang batas, aplikasi memerlukan interaksi manusia untuk persetujuan. Jika tidak, aplikasi disetujui secara otomatis dan pindah ke langkah berikutnya.

Anda akan meniru langkah interaksi manusia dengan menjeda eksekusi alur kerja sampai token tugas dikembalikan. Untuk melakukan ini, Anda akan meneruskan token tugas ke integrasi AWS SDK yang akan Anda gunakan dalam tutorial ini, yaitu Amazon Simple Notification Service. Eksekusi alur kerja akan dijeda sampai menerima token tugas kembali dengan panggilan `SendTaskSuccess`API.

Untuk informasi selengkapnya tentang menggunakan token tugas, lihatTunggu Panggilan Balik dengan Token Tugas.

Karena Anda telah menentukan langkah-langkah untuk persetujuan manusia dan persetujuan otomatis dalam prototipe alur kerja Anda, dalam tutorial ini, Anda terlebih dahulu membuat topik Amazon SNS yang menerima token callback. Kemudian, Anda membuat fungsi Lambda untuk mengimplementasikan fungsionalitas callback. Akhirnya, Anda memperbarui prototipe alur kerja Anda dengan menambahkan rincian integrasi iniLayanan AWS.

Topik

- Langkah 1: Membuat topik Amazon SNS yang menerima token callback
- Langkah 2: Buat fungsi Lambda untuk menangani callback
- Langkah 3: Perbarui alur kerja-tambahkan logika kondisi jika-lain dalam keadaan Choice
- Langkah selanjutnya

Langkah 1: Membuat topik Amazon SNS yang menerima token callback

Untuk mengimplementasikan langkah interaksi manusia, Anda akan memublikasikan topik Amazon Simple Notification Service dan meneruskan token tugas callback ke topik ini. Tugas callback akan menghentikan eksekusi alur kerja sampai token tugas dikembalikan dengan payload.

1. Buka konsol Amazon SNS dan buat tipe topik Standar. Untuk informasi tentang membuat topik, lihat Membuat topik Amazon SNS di Panduan Pengembang Layanan Pemberitahuan Sederhana Amazon.
2. Tentukan nama topik sebagai**`TaskTokenTopic`**.
3. Pastikan untuk menyalin topik ARN dan menyimpannya dalam file teks. Anda memerlukan topik ARN saat menentukan integrasi layanan untuk status Wait for human approval. Berikut ini adalah contoh topik ARN:

   ```
   arn:aws:sns:us-east-2:123456789012:TaskTokenTopic
   ```

4. Buat langganan berbasis email untuk topik tersebut dan kemudian konfirmasikan langganan Anda. Untuk informasi tentang berlangganan topik, lihat Membuat langganan topik di Panduan Pengembang Layanan Pemberitahuan Sederhana Amazon.

Langkah 2: Buat fungsi Lambda untuk menangani callback

Untuk menangani fungsionalitas callback, Anda akan menentukan fungsi Lambda dan menambahkan topik Amazon SNS yang Anda buat di Langkah 1 sebagai pemicu fungsi ini. Saat Anda memublikasikan ke topik Amazon SNS dengan token tugas, fungsi Lambda dipanggil dengan payload pesan yang dipublikasikan.

- Langkah 2.1: Buat fungsi Lambda untuk menangani callback
- Langkah 2.2: Tambahkan topik Amazon SNS sebagai pemicu untuk fungsi Lambda
- Langkah 2.3: Berikan izin yang diperlukan untuk peran IAM fungsi Lambda

Langkah 2.1: Buat fungsi Lambda untuk menangani callback

Dalam fungsi ini, Anda akan memproses permintaan persetujuan batas kredit dan mengembalikan hasil permintaan sebagai berhasil dengan panggilan `SendTaskSuccess`API. Fungsi Lambda ini juga akan mengembalikan token tugas yang diterimanya dari topik Amazon SNS.

Untuk mempermudah, fungsi Lambda yang digunakan untuk langkah interaksi manusia secara otomatis menyetujui tugas apa pun dan mengembalikan token tugas dengan `SendTaskSuccess` panggilan API. Anda dapat menamai fungsi Lambda sebagai. **`callback-human-approval`**

1. Di tab atau jendela baru, buka konsol Lambda dan buat fungsi Node.js 16.x Lambda berjudul. **`callback-human-approval`** Untuk informasi tentang membuat fungsi Lambda menggunakan konsol, lihat Membuat fungsi Lambda di konsol di Panduan Pengembang. AWS Lambda
2. Pada callback-human-approvalhalaman, ganti kode yang ada di area sumber Kode dengan kode berikut.

```
// Sample Lambda function that will automatically approve any task whenever a
 message is published to an Amazon SNS topic by Step Functions.

console.log('Loading function');
const AWS = require('aws-sdk');
const resultMessage = "Successful";

exports.handler = async (event, context) => {
    const stepfunctions = new AWS.StepFunctions();

    let message = JSON.parse(event.Records[0].Sns.Message);
    let taskToken = message.TaskToken;
```

```
    console.log('Message received from SNS:', message);
    console.log('Task token: ', taskToken);

    // Return task token to Step Functions

    let params = {
        output: JSON.stringify(resultMessage),
        taskToken: taskToken
    };

    console.log('JSON Returned to Step Functions: ', params);
    let myResult = await stepfunctions.sendTaskSuccess(params).promise();
    console.log('State machine - callback completed..');

    return myResult;

};
```

3. Jauhkan jendela ini terbuka dan lakukan langkah-langkah di bagian berikutnya untuk tindakan lebih lanjut.

Langkah 2.2: Tambahkan topik Amazon SNS sebagai pemicu untuk fungsi Lambda

Ketika Anda menambahkan topik Amazon SNS yang Anda buat di Langkah 1 dari tutorial ini sebagai pemicu untuk fungsi Lambda yang Anda buat di Langkah 2.1 dari tutorial ini, fungsi Lambda dipicu setiap kali Anda mempublikasikan ke topik Amazon SNS. Saat Anda memublikasikan ke topik Amazon SNS dengan token tugas, fungsi Lambda dipanggil dengan payload pesan yang dipublikasikan. Untuk informasi selengkapnya tentang mengonfigurasi pemicu untuk fungsi Lambda, lihat Mengonfigurasi pemicu di Panduan Pengembang. AWS Lambda

1. Di bagian Ringkasan fungsi fungsi fungsi `callback-human-approval` Lambda, pilih Tambah pemicu.
2. Dari daftar drop-down pemicu, pilih SNS sebagai pemicu.
3. Untuk topik SNS, mulailah mengetik nama topik Amazon SNS yang Anda buat di Langkah 1 dari tutorial ini, dan pilih dari daftar dropdown yang muncul.
4. Pilih Tambahkan.
5. Jauhkan jendela ini terbuka dan lakukan langkah-langkah di bagian berikutnya untuk tindakan lebih lanjut.

Langkah 2.3: Berikan izin yang diperlukan untuk peran IAM fungsi Lambda

Anda harus memberikan fungsi `callback-human-approval` Lambda izin untuk mengakses Step Functions untuk mengembalikan token tugas bersama dengan panggilan API. `SendTaskSucess`

1. Pada callback-human-approvalhalaman, pilih tab Konfigurasi, lalu pilih Izin.
2. Di bawah Peran Eksekusi, pilih Nama peran untuk menavigasi ke halaman Peran AWS Identity and Access Management konsol.
3. Untuk menambahkan izin yang diperlukan, pilih Tambahkan izin, lalu pilih Lampirkan kebijakan.
4. Di kotak pencarian, ketik **`AWSStepFunctions`** dan kemudian tekan Enter.
5. Pilih AWSStepFunctionsFullAccesslalu gulir ke bawah untuk memilih Lampirkan kebijakan. Ini menambahkan kebijakan yang berisi izin yang diperlukan untuk peran fungsi `callback-human-approval` Lambda.

Langkah 3: Perbarui alur kerja-tambahkan logika kondisi jika-lain dalam keadaan Choice

Di konsol Step Functions, tentukan logika kondisional untuk alur kerja Anda menggunakan state. `Choice` Jika output yang dikembalikan oleh fungsi `RandomNumberforCredit` Lambda kurang dari 5000, kredit yang diminta disetujui secara otomatis. Jika output kembali lebih besar dari atau sama dengan 5000, eksekusi alur kerja hasil untuk langkah interaksi manusia untuk persetujuan batas kredit.

Dalam `Choice` keadaan, Anda menggunakan operator perbandingan untuk membandingkan variabel input dengan nilai tertentu. Anda dapat menentukan variabel input sebagai input eksekusi saat memulai eksekusi mesin status atau menggunakan output dari langkah sebelumnya sebagai input untuk langkah saat ini. Secara default, output dari langkah disimpan dalam variabel yang disebut`Payload`. Untuk menggunakan nilai `Payload` variabel untuk perbandingan di `Choice` negara bagian, gunakan $ sintaks seperti yang ditunjukkan dalam prosedur berikut.

Untuk informasi tentang bagaimana informasi mengalir dari satu negara ke negara lain dan menentukan input dan output dalam alur kerja Anda, lihat Tutorial 7: Konfigurasikan input dan output dan. Pengolahan Input dan Output di Step Functions

> **Note**
>
> Jika `Choice` negara menggunakan variabel masukan yang ditentukan dalam input eksekusi mesin negara untuk perbandingan, gunakan `$.variable_name` sintaks untuk melakukan perbandingan. Misalnya, untuk membandingkan variabel, sepertimyAge, gunakan sintaks`$.myAge`.

Karena dalam langkah ini, `Choice` negara akan menerima masukan dari status batas kredit Get, Anda akan menggunakan $ sintaks untuk konfigurasi `Choice` negara. Untuk mengeksplorasi bagaimana hasil eksekusi mesin negara berbeda ketika Anda menggunakan `$.variable_name` sintaks dalam konfigurasi `Choice` status untuk merujuk ke output dari langkah sebelumnya, lihat Debugging galat status Choice jalur tidak valid bagian dalam Tutorial 8.

Untuk menambahkan logika kondisi jika-lain menggunakan negara **`Choice`**

1. Buka jendela konsol Step Functions yang berisi prototipe alur kerja yang Anda buat. Tutorial 1: Buat prototipe untuk mesin negara Anda
2. Pilih Kredit yang diterapkan >= 5000? state dan di tab Configuration, tentukan logika kondisional sebagai berikut:
   a. Di bawah Aturan Pilihan, pilih ikon Edit di ubin Aturan #1 untuk menentukan aturan pilihan pertama.
   b. Pilih Tambahkan kondisi.
   c. Di kotak dialog Conditions for rule #1, untuk Variabel, masukkan**$**.
   d. Untuk Operator, pilih kurang dari.
   e. Untuk Nilai, pilih Jumlah konstan, dan kemudian masukkan **`5000`** di bidang di sebelah Nilai daftar dropdown.
   f. Pilih Simpan kondisi.
   g. Untuk Kemudian negara berikutnya adalah: daftar dropdown, pilih Auto-approve limit.
   h. Pilih Tambahkan aturan pilihan baru, dan kemudian tentukan aturan pilihan kedua ketika jumlah kredit lebih besar dari atau sama dengan 5000 dengan mengulangi substeps 2.b hingga 2.f. Untuk Operator, pilih lebih besar dari atau sama dengan.
   i. Untuk Kemudian negara berikutnya adalah: daftar dropdown, pilih Tunggu persetujuan manusia.

j. Dalam Aturan default kotak, pilih ikon Edit untuk menentukan aturan pilihan default, dan kemudian pilih Tunggu persetujuan manusia dari Status default daftar dropdown. Anda menentukan aturan Default untuk menentukan status berikutnya untuk transisi ke jika tidak ada kondisi negara Choice mengevaluasi ke benar atau salah.

3. Konfigurasikan status Tunggu persetujuan manusia sebagai berikut:

   a. Di tab Konfigurasi, untuk Topik, mulailah mengetik nama topik Amazon SNS TaskTokenTopic, dan pilih nama seperti yang muncul di daftar dropdown.

   b. Untuk Pesan, pilih Masukkan pesan dari daftar dropdown. Di bidang Pesan, Anda menentukan pesan yang ingin Anda publikasikan ke topik Amazon SNS. Untuk tutorial ini, Anda mempublikasikan token tugas sebagai pesan.

      Token tugas memungkinkan Anda menjeda alur kerja Step Functions tipe Standar sampai proses eksternal selesai dan token tugas dikembalikan. Saat Anda menentukan status Tugas sebagai tugas callback dengan menentukan pola integrasi `.waitForTaskToken` layanan, token tugas dibuat dan ditempatkan di objek konteks saat tugas dimulai. Objek konteks adalah struktur JSON internal yang tersedia selama eksekusi, dan berisi informasi tentang mesin negara Anda dan pelaksanaannya. Untuk informasi lebih lanjut tentang objek konteks, lihatObjek konteks.

   c. Di kotak yang muncul, masukkan pesan berikut sebagai:

```
{
  "TaskToken.$": "$$.Task.Token"
}
```

   d. Pilih kotak centang Tunggu callback.

   e. Pilih Selesai di kotak dialog yang muncul.

4. Jauhkan jendela ini terbuka dan lanjutkan ke tutorial berikutnya untuk tindakan lebih lanjut.

Langkah selanjutnya

Dalam tutorial berikutnya, Anda akan belajar bagaimana melakukan beberapa tugas secara paralel.

Tutorial 4: Tentukan beberapa tugas untuk dilakukan secara paralel

Sejauh ini Anda telah belajar cara menjalankan alur kerja secara berurutan. Namun, Anda dapat menjalankan dua atau lebih langkah secara paralel menggunakan `Parallel` state. Sebuah `Parallel` negara menyebabkan interpreter untuk mengeksekusi setiap cabang secara bersamaan.

Kedua cabang dalam `Parallel` keadaan menerima masukan yang sama, tetapi setiap cabang memproses bagian-bagian masukan khusus untuk itu. Langkah Fungsi menunggu sampai setiap cabang selesai mengeksekusi sebelum melanjutkan ke langkah berikutnya.

Dalam tutorial ini, Anda menggunakan keadaan Paralel untuk secara bersamaan memeriksa identitas dan alamat pelamar.

Topik

- Langkah 1: Buat fungsi Lambda untuk melakukan pemeriksaan yang diperlukan
- Langkah 2: Perbarui alur kerja - Tambahkan tugas paralel yang akan dilakukan

Langkah 1: Buat fungsi Lambda untuk melakukan pemeriksaan yang diperlukan

Alur kerja aplikasi kartu kredit ini memanggil dua fungsi Lambda di dalam keadaan Paralel untuk memeriksa identitas dan alamat pemohon. Pemeriksaan ini dilakukan secara bersamaan menggunakan keadaan Paralel. Mesin negara menyelesaikan eksekusi hanya setelah kedua cabang paralel telah selesai mengeksekusi.

Untuk membuat fungsi Check-Identity dan Check-Address Lambda

1. Di tab atau jendela baru, buka konsol Lambda dan buat dua fungsi Node.js 16.x Lambda berjudul dan. `check-identity` `check-address` Untuk informasi tentang membuat fungsi Lambda menggunakan konsol, lihat Membuat fungsi Lambda di konsol di Panduan Pengembang. AWS Lambda
2. Buka halaman fungsi check-identity dan ganti kode yang ada di area sumber Kode dengan kode berikut:

```
const ssnRegex = /^\d{3}-?\d{2}-?\d{4}$/;
const emailRegex = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/;

class ValidationError extends Error {
```

```
    constructor(message) {
        super(message);
        this.name = "CustomValidationError";
    }
}

exports.handler = async (event) => {
    const {
        ssn,
        email
    } = event;
    console.log(`SSN: ${ssn} and email: ${email}`);

    const approved = ssnRegex.test(ssn) && emailRegex.test(email);

    if (!approved) {
        throw new ValidationError("Check Identity Validation Failed");
    }

    return {
        statusCode: 200,
        body: JSON.stringify({
            approved,
            message: `Identity validation ${approved ? 'passed' : 'failed'}`
        })
    }
};
```

3. Buka halaman fungsi check-address dan ganti kode yang ada di area sumber Kode dengan kode berikut:

```
class ValidationError extends Error {
  constructor(message) {
      super(message);
      this.name = "CustomAddressValidationError";
  }
}

exports.handler = async event => {
  const {
      street,
      city,
      state,
```

```
        zip
    } = event;
    console.log(`Address information: ${street}, ${city}, ${state} - ${zip}`);

    const approved = [street, city, state, zip].every(i => i?.trim().length > 0);

    if (!approved) {
        throw new ValidationError("Check Address Validation Failed");
    }

    return {
        statusCode: 200,
        body: JSON.stringify({
            approved,
            message: `Address validation ${ approved ? 'passed' : 'failed'}`
        })
    }
};
```

4. Untuk kedua fungsi Lambda, dari bagian Ringkasan fungsi, salin Amazon Resource Names (ARN) masing-masing dan simpan dalam file teks. Anda memerlukan ARN fungsi saat menentukan integrasi layanan untuk status identitas dan alamat Verifikasi pemohon. Berikut ini adalah contoh ARN:

```
arn:aws:lambda:us-east-2:123456789012:function:HelloWorld
```

Langkah 2: Perbarui alur kerja - Tambahkan tugas paralel yang akan dilakukan

Di konsol Step Functions, Anda akan memperbarui alur kerja Anda untuk menentukan integrasi layanan dengan fungsi check-identity dan check-address Lambda yang Anda buat di Langkah 1.

Untuk menambahkan tugas paralel dalam alur kerja

1. Buka jendela konsol Step Functions yang berisi prototipe alur kerja yang Anda buat. Tutorial 1: Buat prototipe untuk mesin negara Anda
2. Pilih status Verifikasi identitas, dan di tab Konfigurasi, lakukan hal berikut:

    a. Untuk tipe Integrasi, pertahankan pilihan default Optimized.

> **Note**
>
> Menggunakan Step Functions, Anda dapat mengintegrasikan dengan yang lain Layanan AWS dan mengatur mereka dalam alur kerja Anda. Untuk informasi selengkapnya tentang integrasi layanan dan jenisnya, lihat Menggunakan AWS Step Functions dengan layanan lain

b. Untuk Nama fungsi, pilih fungsi Lambda check-identitas dari daftar dropdown.

c. Untuk Payload, pilih Enter payload dan kemudian ganti contoh payload dengan yang berikut sebagai payload:

```
{
    "email": "janedoe@example.com",
    "ssn": "012-00-0000"
}
```

3. Pilih status Verifikasi alamat, dan di tab Konfigurasi, lakukan hal berikut:

a. Untuk tipe Integrasi, pertahankan pilihan default Optimized.

b. Untuk Nama fungsi, pilih fungsi Lambda check-address dari daftar dropdown.

c. Untuk Payload, pilih Enter payload dan kemudian ganti contoh payload dengan yang berikut sebagai payload:

```
{
    "street": "123 Any St",
    "city": "Any Town",
    "state": "AT",
    "zip": "01000"
}
```

4. Pilih Selanjutnya.

Tutorial 5: Bersamaan mengulangi koleksi item

Dalam tutorial sebelumnya, Anda belajar cara menjalankan cabang langkah terpisah secara paralel menggunakan `Parallel` status. Dengan menggunakan `Map` status, Anda dapat menjalankan serangkaian langkah alur kerja untuk setiap item dalam kumpulan data. Iterasi `Map` status berjalan secara paralel, yang memungkinkan untuk memproses kumpulan data dengan cepat.

Dengan menyertakan `Map` status dalam alur kerja Anda, Anda dapat melakukan tugas, seperti pemrosesan data, menggunakan salah satu dari duaMode pemrosesan status peta: Mode sebaris dan mode Terdistribusi. Untuk mengonfigurasi `Map` status, Anda menentukan`ItemProcessor`, yang berisi objek JSON yang menentukan mode pemrosesan `Map` status dan definisinya. Dalam tutorial ini, Anda menjalankan `Map` status dalam mode Inline default, yang mendukung hingga 40 iterasi bersamaan. Saat Anda menjalankan `Map` status dalam mode Terdistribusi, ini mendukung hingga 10.000 eksekusi alur kerja anak paralel.

Ketika eksekusi alur kerja Anda memasuki `Map` status, itu akan mengulangi array JSON yang ditentukan dalam input status. Untuk setiap item array, iterasi yang sesuai berjalan dalam konteks alur kerja yang berisi status. `Map` Ketika semua iterasi selesai, `Map` status akan mengembalikan array yang berisi output untuk setiap item yang diproses oleh. `ItemProcessor`

Dalam tutorial ini, Anda mempelajari cara menggunakan `Map` status dalam mode Inline untuk mengambil skor kredit pemohon dengan mengulangi serangkaian biro kredit. Untuk melakukan ini, pertama-tama Anda mengambil nama semua biro kredit yang disimpan dalam tabel Amazon DynamoDB, dan kemudian menggunakan `Map` negara untuk memutar melalui daftar biro kredit untuk mengambil skor kredit pemohon yang dilaporkan oleh masing-masing biro ini.

Topik

- Langkah 1: Buat tabel DynamoDB untuk menyimpan nama semua biro kredit
- Langkah 2: Perbarui mesin negara - Ambil hasil dari tabel DynamoDB
- Langkah 3: Buat fungsi Lambda yang mengembalikan skor kredit untuk semua biro kredit
- Langkah 4: Perbarui mesin status - tambahkan status Peta untuk mengambil skor kredit secara berulang

Langkah 1: Buat tabel DynamoDB untuk menyimpan nama semua biro kredit

Pada langkah ini, Anda membuat tabel bernama **`GetCreditBureau`** menggunakan konsol DynamoDB. Tabel menggunakan atribut string Nama sebagai kunci Partition. Dalam tabel ini, Anda menyimpan nama semua biro kredit tempat Anda ingin mengambil skor kredit pemohon.

1. Masuk ke AWS Management Console dan buka konsol DynamoDB di https://console.aws.amazon.com/dynamodb/.
2. Di panel navigasi di konsol, pilih Tabel, lalu pilih Buat tabel.

3. Masukkan detail tabel sebagai berikut:

   a. Untuk Nama tabel, masukkan **`GetCreditBureau`**.

   b. Untuk tombol Partition, masukkan**`Name`**.

   c. Simpan pilihan default, dan pilih Buat tabel.

4. Setelah tabel Anda dibuat, dalam daftar Tabel, pilih GetCreditBureautabel.
5. Pilih Tindakan, lalu pilih Buat item.
6. Untuk Nilai, masukkan nama biro kredit. Sebagai contoh, **`CredTrack`**.
7. Pilih Buat item.
8. Ulangi proses ini dan buat item untuk nama biro kredit lainnya. Misalnya, **`KapFinn`** dan **`CapTrust`**.

Langkah 2: Perbarui mesin negara - Ambil hasil dari tabel DynamoDB

Di konsol Step Functions, Anda akan menambahkan `Task` status dan menggunakan integrasi AWS SDK untuk mengambil nama biro kredit dari tabel DynamoDB yang Anda buat di Langkah 1. Anda akan menggunakan output dari langkah ini sebagai masukan untuk `Map` status yang akan Anda tambahkan nanti dalam alur kerja Anda dalam tutorial ini.

1. Buka mesin CreditCardWorkflownegara untuk memperbaruinya.
2. Pilih daftar Dapatkan negara biro kredit.
3. Untuk Parameter API, tentukan nilai nama Tabel sebagai**`GetCreditBureau`**.

Langkah 3: Buat fungsi Lambda yang mengembalikan skor kredit untuk semua biro kredit

Pada langkah ini, Anda membuat fungsi Lambda yang menerima nama semua biro kredit sebagai masukan, dan mengembalikan skor kredit pemohon untuk masing-masing biro kredit ini. Fungsi Lambda ini akan dipanggil dari `Map` status yang akan Anda tambahkan dalam alur kerja Anda di Langkah 4 dari tutorial ini.

1. Buat fungsi Lambda Node.js 16.x dan beri nama. **`get-credit-score`**
2. Pada halaman berjudul get-credit-score, tempel kode berikut ke area sumber Kode.

```
function getScore(arr) {
```

```
  let temp;
  let i = Math.floor((Math.random() * arr.length));
  temp = arr[i];
  console.log(i);
  console.log(temp);
  return temp;
}

const arrScores = [700, 820, 640, 460, 726, 850, 694, 721, 556];

exports.handler = (event, context, callback) => {
  let creditScore = getScore(arrScores);
  callback(null, "Credit score pulled is: " + creditScore + ".");
};
```

3. Menyebarkan fungsi Lambda.

Langkah 4: Perbarui mesin status - tambahkan status Peta untuk mengambil skor kredit secara berulang

Di konsol Step Functions, Anda menambahkan `Map` status yang memanggil fungsi get-credit-scoreLambda untuk memeriksa skor kredit pemohon untuk semua biro kredit yang dikembalikan oleh daftar Dapatkan status biro kredit.

1. Buka mesin CreditCardWorkflownegara untuk memperbaruinya.
2. Pilih Dapatkan skor dari semua negara biro kredit.
3. Di tab Konfigurasi, pilih Menyediakan jalur ke array item dan kemudian masukkan**`$.Items`**.
4. Pilih Dapatkan semua skor langkah di dalam `Map` negara bagian.
5. Di tab Konfigurasi, pastikan untuk jenis Integrasi, Dioptimalkan dipilih.
6. Untuk nama Fungsi, mulailah mengetik nama fungsi get-credit-scoreLambda dan pilih dari daftar dropdown yang muncul.
7. Untuk Payload, pilih No payload.

Tutorial 6: Simpan alur kerja dan jalankan mesin negara

Sekarang setelah Anda mengonfigurasi sumber daya dari semua yang Layanan AWS Anda gunakan dalam prototipe alur kerja, Anda dapat menyimpannya sebagai mesin status Step Functions dan mulai menjalankannya.

Topik

- Langkah 1: Tinjau definisi mesin status yang dihasilkan secara otomatis dan simpan mesin status
- Langkah 2: Tambahkan kebijakan IAM yang tersisa
- Langkah 3: Jalankan mesin negara

Langkah 1: Tinjau definisi mesin status yang dihasilkan secara otomatis dan simpan mesin status

Saat Anda menyeret dan melepas status dari tab Flow ke kanvas di Workflow Studio untuk membangun prototipe alur kerja, Step Functions secara otomatis menyusun definisi Amazon States Language (ASL) alur kerja Anda secara real-time. Anda dapat mengedit definisi ini seperti yang diperlukan dalamEditor kode.

Untuk meninjau definisi ASL dan menyimpan mesin negara

1. (Opsional) Pilih Definisi pada Inspector untuk melihat definisi state machine Amazon States Language (ASL), yang secara otomatis dihasilkan berdasarkan pilihan Anda di tab Actions and Flow dan panel Inspector.

   > **Tip**
   >
   > Untuk mengedit definisi, Anda dapat membuka editor kode dengan memilih Kode di bagian atas halaman. Untuk tutorial ini, lanjutkan dengan definisi yang dibuat secara otomatis.

2. Tentukan nama untuk mesin negara Anda. Untuk melakukan ini, pilih ikon edit di sebelah nama mesin status default MyStateMachine. Kemudian, dalam konfigurasi mesin Negara, tentukan nama di kotak Nama mesin Negara.

   Untuk tutorial ini, masukkan nama **`CreditCardWorkflow`**.

3. (Opsional) Dalam konfigurasi mesin State, tentukan pengaturan alur kerja lainnya, seperti jenis mesin status dan peran pelaksanaannya.

   Untuk tutorial ini, simpan semua pilihan default di pengaturan mesin State.

   > **Note**
   >
   > (Opsional) Step Functions secara otomatis membuat peran eksekusi untuk mesin status dengan hak istimewa paling sedikit yang diperlukan untuk menjalankan fungsi `RandomNumberforCredit` Lambda dan memublikasikan ke topik Amazon SNS.

   Jika sebelumnya Anda telah membuat peran IAM dengan izin yang benar untuk mesin status dan ingin menggunakannya, di Izin, pilih Pilih peran yang ada, lalu pilih peran dari daftar. Atau pilih Masukkan peran ARN dan kemudian berikan ARN untuk peran IAM itu.

4. Dalam kotak dialog Konfirmasi pembuatan peran, pilih Konfirmasi untuk melanjutkan.

   Anda juga dapat memilih Lihat pengaturan peran untuk kembali ke konfigurasi mesin Status.

   > **Note**
   >
   > Jika Anda menghapus IAM role yang Step Functions buat, Step Functions tidak dapat membuatnya kembali nanti. Demikian pula, jika Anda mengubah peran (misalnya, dengan menghapus Step Functions dari principal dalam kebijakan IAM), Step Functions tidak dapat memulihkan pengaturan aslinya nanti.

Langkah 2: Tambahkan kebijakan IAM yang tersisa

Karena Step Functions tidak otomatis menghasilkan izin untuk memanggil fungsi Lambda yang digunakan di `Parallel` negara bagian, Anda perlu menambahkan kebijakan yang diperlukan.

Untuk menambahkan kebijakan yang tersisa

1. Pada CreditCardWorkflowhalaman, pilih peran IAM untuk mesin status Anda untuk menavigasi ke konsol IAM. Anda akan menambahkan izin yang diperlukan untuk fungsi Lambda yang tersisa di halaman ini.
2. Pilih Tambahkan izin, lalu pilih Lampirkan kebijakan.

3. Di kotak pencarian, ketik lalu **`AWSLambdaRole`** tekan Enter.
4. Pilih AWSLambdaRoledan kemudian pilih Lampirkan kebijakan. Kebijakan ini sekarang ditambahkan ke peran eksekusi mesin status Anda. Kebijakan ini memungkinkan Anda menjalankan fungsi Lambda apa pun di mesin status Anda.

Langkah 3: Jalankan mesin negara

Eksekusi mesin status adalah instans tempat Anda menjalankan alur kerja untuk melakukan tugas.

Untuk mengeksekusi mesin negara

1. Pada CreditCardWorkflowhalaman, pilih Mulai eksekusi.

   Kotak dialog Mulai eksekusi ditampilkan.
2. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

   a. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions secara otomatis menghasilkan nama eksekusi yang unik.

   > **Note**
   >
   > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, aktivitas, dan label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

   > **Note**
   >
   > Anda tidak perlu memberikan masukan apa pun untuk menjalankan mesin status ini. Tetapi Anda dapat menentukan input eksekusi, jika diperlukan, di area Input dari kotak dialog Mulai eksekusi untuk mesin negara lainnya. Untuk contoh cara memberikan input eksekusi ke mesin status, lihat Langkah 4: Memulai eksekusi baru dari tutorial Belajar menggunakan AWS Step Functions Workflow Studio.

   b. Pilih Mulai Eksekusi.

3. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

   Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Tutorial 7: Konfigurasikan input dan output

Sebuah eksekusi Step Functions menerima teks JSON sebagai input dan meneruskan input yang ke status pertama dalam alur kerja. Status individu dalam alur kerja menerima data JSON sebagai input dan biasanya meneruskan data JSON sebagai output ke status berikutnya. Secara default, data melewati dari satu negara ke keadaan berikutnya dalam alur kerja kecuali Anda telah mengkonfigurasi input dan/atau output untuk satu atau lebih negara dalam alur kerja. Memahami bagaimana informasi mengalir dari negara ke negara lain, dan mempelajari cara memfilter dan memanipulasi data ini, adalah kunci untuk merancang dan menerapkan alur kerja secara efektif dalam Step Functions.

Langkah Fungsi menyediakan beberapa filter untuk mengontrol input dan output aliran data antara negara. Filter berikut tersedia untuk digunakan dalam alur kerja Anda:

> **Note**
>
> Berdasarkan kasus penggunaan Anda, Anda mungkin tidak perlu menerapkan semua filter ini dalam alur kerja Anda.

`InputPath`

Memilih bagian APA dari seluruh muatan masukan yang akan digunakan sebagai masukan tugas. Jika Anda menentukan bidang ini, Step Functions pertama-tama menerapkan bidang ini.

`Parameter`

Menentukan BAGAIMANA input akan terlihat seperti sebelum memohon tugas. Dengan `Parameters` lapangan, Anda dapat membuat koleksi pasangan kunci-nilai yang dilewatkan

sebagai masukan ke Layanan AWSintegrasi, seperti fungsi. AWS Lambda Nilai-nilai ini dapat statis, atau dipilih secara dinamis baik dari input negara atau objek konteks alur kerja.

`ResultSelector`

Menentukan APA untuk memilih dari output tugas. Dengan `ResultSelector` bidang, Anda dapat membuat koleksi pasangan kunci-nilai yang menggantikan hasil negara dan meneruskan koleksi itu. `ResultPath`

`ResultPath`

Menentukan WHERE untuk menempatkan output tugas. Gunakan `ResultPath` untuk menentukan apakah output dari suatu negara adalah salinan dari input, hasil yang dihasilkannya, atau kombinasi keduanya.

`OutputPath`

Menentukan APA yang akan dikirim ke keadaan berikutnya. Dengan `OutputPath` itu, Anda dapat memfilter informasi yang tidak diinginkan, dan hanya meneruskan sebagian data JSON yang Anda pedulikan.

> **Tip**
>
> The `Parameters` dan `ResultSelector` filter bekerja dengan membangun JSON, sedangkan `InputPath` dan `OutputPath` filter bekerja dengan menyaring node tertentu dalam objek data JSON, dan `ResultPath` filter bekerja dengan menciptakan bidang di mana output dapat ditambahkan.

Dalam tutorial ini, Anda belajar bagaimana melakukan tugas-tugas berikut:

- Pilih bagian tertentu dari input mentah menggunakan InputPath filter
- Memanipulasi masukan yang dipilih menggunakan filter Parameter
- Mengkonfigurasi output menggunakanResultSelector,ResultPath, dan OutputPath filter

Untuk informasi selengkapnya tentang mengonfigurasi input dan output dalam alur kerja Anda, lihat. Pengolahan Input dan Output di Step Functions

Pilih bagian tertentu dari input mentah menggunakan InputPath filter

Gunakan `InputPath` filter untuk memilih bagian tertentu dari muatan input.

Jika Anda tidak menentukan`InputPath`, nilainya default$, yang menyebabkan tugas status merujuk ke seluruh input mentah, bukan bagian tertentu.

Untuk mempelajari cara menggunakan InputPath filter, lakukan langkah-langkah berikut:

- Langkah 1: Buat mesin status
- Langkah 2: Jalankan mesin negara
- Langkah 3: Gunakan InputPath filter untuk memilih bagian tertentu dari input eksekusi

Langkah 1: Buat mesin status

> **Important**
>
> Pastikan mesin status Anda berada di bawah AWS akun dan Wilayah yang sama dengan fungsi Lambda yang Anda buat sebelumnya.

1. Gunakan contoh `Parallel` status yang Anda pelajari di Tutorial 4 untuk membuat mesin status baru. Pastikan prototipe alur kerja Anda terlihat mirip dengan prototipe berikut.
2. Konfigurasikan integrasi untuk fungsi `check-identity` dan `check-address` Lambda. Untuk informasi tentang membuat fungsi Lambda dan menggunakannya di mesin status Anda, lihat Langkah 1: Buat fungsi Lambda untuk melakukan pemeriksaan yang diperlukan dan. Langkah 2: Perbarui alur kerja - Tambahkan tugas paralel yang akan dilakukan
3. Untuk Payload, pastikan Anda menyimpan pilihan default Use state input as payload.
4. Pilih Berikutnya dan kemudian lakukan langkah 1 hingga 3 di Langkah 1: Simpan mesin negara Tutorial 5 untuk membuat mesin status baru. Untuk tutorial ini, beri nama mesin negara Anda**`WorkflowInputOutput`**.

Langkah 2: Jalankan mesin negara

1. Pada WorkflowInputOutputhalaman, pilih Mulai eksekusi.
2. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions secara otomatis menghasilkan nama eksekusi yang unik.

> **Note**
>
> Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, aktivitas, dan label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

3. Di area Input, tambahkan data JSON berikut sebagai input eksekusi.

```
{
  "data": {
    "firstname": "Jane",
    "lastname": "Doe",
    "identity": {
      "email": "jdoe@example.com",
      "ssn": "123-45-6789"
    },
    "address": {
      "street": "123 Main St",
      "city": "Columbus",
      "state": "OH",
      "zip": "43219"
    }
  }
}
```

4. Pilih Mulai Eksekusi.
5. Eksekusi mesin status menghasilkan kesalahan karena Anda belum menentukan bagian mana dari input eksekusi `check-identity` dan fungsi `check-address` Lambda yang harus digunakan untuk melakukan verifikasi identitas dan alamat yang diperlukan.
6. Lanjutkan ke Langkah 3 dari tutorial ini untuk memperbaiki kesalahan.

Langkah 3: Gunakan **`InputPath`** filter untuk memilih bagian tertentu dari input eksekusi

1. Pada halaman Rincian Eksekusi, pilih Edit state machine.

2. Untuk memverifikasi identitas pemohon sebagaimana disebutkan dalam input eksekusi yang disediakanLangkah 2: Jalankan mesin negara, edit definisi tugas Verifikasi identitas sebagai berikut:

```
...
{
    "StartAt": "Verify identity",
    "States": {
      "Verify identity": {
        "Type": "Task",
        "Resource": "arn:aws:states:::lambda:invoke",
        "InputPath": "$.data.identity",
        "Parameters": {
          "Payload.$": "$",
          "FunctionName": "arn:aws:lambda:us-east-2:123456789012:function:check-
identity:$LATEST"
        },
        "End": true
      }
    }
  }
  ...
```

Akibatnya, data JSON berikut menjadi tersedia sebagai input untuk `check-identity` fungsi tersebut.

```
{
   "email": "jdoe@example.com",
   "ssn": "123-45-6789"
 }
```

3. Untuk memverifikasi alamat pemohon sebagaimana disebutkan dalam input eksekusi, edit definisi `Verify address` tugas sebagai berikut:

```
...
{
    "StartAt": "Verify address",
    "States": {
      "Verify address": {
        "Type": "Task",
        "Resource": "arn:aws:states:::lambda:invoke",
        "InputPath": "$.data.address",
```

```
        "Parameters": {
          "Payload.$": "$",
          "FunctionName": "arn:aws:lambda:us-east-1:123456789012:function:check-
address:$LATEST"
        },
        "End": true
      }
    }
  }
  ...
```

Akibatnya, data JSON berikut menjadi tersedia sebagai input untuk `check-address` fungsi tersebut.

```
{
    "street": "123 Main St",
    "city": "Columbus",
    "state": "OH",
    "zip": "43219"
}
```

4. Pilih Mulai Eksekusi. Eksekusi mesin status sekarang berhasil diselesaikan.

Memanipulasi masukan yang dipilih menggunakan filter Parameter

Sementara `InputPath` filter membantu Anda membatasi input JSON mentah yang Anda berikan, menggunakan `Parameters` filter, Anda dapat meneruskan kumpulan pasangan kunci-nilai sebagai input. Pasangan kunci-nilai ini dapat berupa nilai statis yang Anda tentukan dalam definisi mesin negara Anda, atau nilai yang dipilih dari input mentah menggunakan. `InputPath`

Dalam alur kerja Anda, `Parameters` diterapkan setelah`InputPath`. `Parameters`membantu Anda menentukan bagaimana tugas yang mendasarinya menerima payload inputnya. Misalnya, jika fungsi `check-address` Lambda menerima parameter string sebagai input, bukan data JSON, Anda dapat menggunakan `Parameters` filter untuk mengubah input.

Pada contoh berikut, `Parameters` filter menerima input yang Anda pilih gunakan `InputPath` Langkah 3: Gunakan `InputPath` filter untuk memilih bagian tertentu dari input eksekusi dan menerapkan fungsi intrinsik `States.Format` pada item input untuk membuat string yang disebut. `addressString` Fungsi intrinsik membantu Anda melakukan operasi pemrosesan data dasar pada input yang diberikan. Untuk informasi selengkapnya, lihat Fungsi intrinsik.

```
"Parameters": {
    "addressString.$": "States.Format('{}. {}, {} - {}', $.street, $.city, $.state,
 $.zip)"
  }
```

Akibatnya, string berikut akan dibuat dan disediakan untuk fungsi `check-address` Lambda sebagai masukan.

```
{
    "addressString": "123 Main St. Columbus, OH - 43219"
}
```

Mengkonfigurasi output menggunakanResultSelector,ResultPath, dan OutputPath filter

Ketika fungsi `check-address` Lambda dipanggil di mesin WorkflowInputOutputstate, fungsi mengembalikan muatan keluaran setelah melakukan verifikasi alamat. Pada halaman Detail Eksekusi, pilih langkah Verifikasi alamat dan lihat payload keluaran di dalam hasil Tugas di Detail langkah panel.

```
{
    "ExecutedVersion": "$LATEST",
    "Payload": {
      "statusCode": 200,
      "body": "{\"approved\":true,\"message\":\"identity validation passed\"}"
    },
    "SdkHttpMetadata": {
      "AllHttpHeaders": {
        "X-Amz-Executed-Version": [
          "$LATEST"
        ],
        ...
        ...
    "StatusCode": 200
  }
```

Menggunakan ResultSelector

Sekarang jika Anda perlu memberikan hasil pemeriksaan verifikasi identitas dan alamat ke status berikut dalam alur kerja Anda, Anda dapat memilih node payload.body dalam output JSON dan

menggunakan fungsi `StringToJson` intrinsik dalam `ResultSelector` filter untuk memformat data sesuai kebutuhan.

`ResultSelector`memilih apa yang dibutuhkan dari output tugas. Pada contoh berikut, **`ResultSelector`** mengambil string di $.payload.body dan menerapkan fungsi **`States.StringToJson`** intrinsik untuk mengkonversi string ke JSON dan menempatkan JSON yang dihasilkan dalam node identitas.

```
"ResultSelector": {
    "identity.$": "States.StringToJson($.Payload.body)"
  }
```

Akibatnya, data JSON berikut dibuat.

```
{
    "identity": {
      "approved": true,
      "message": "Identity validation passed"
    }
  }
```

Saat Anda bekerja dengan filter input dan output ini, Anda juga dapat menemukan kesalahan runtime yang timbul karena menentukan ekspresi jalur JSON yang tidak valid. Untuk informasi selengkapnya, lihat .

Menggunakan ResultPath

Anda dapat menentukan lokasi di payload input awal untuk menyimpan hasil pemrosesan tugas negara menggunakan `ResultPath` bidang. Jika Anda tidak menentukan`ResultPath`, nilainya default$, yang menyebabkan payload input awal diganti dengan hasil tugas mentah. Jika Anda menentukan `ResultPath` sebagai`null`, hasil mentah dibuang dan muatan masukan awal menjadi output yang efektif.

Jika Anda menerapkan `ResultPath` bidang pada data JSON yang dibuat menggunakan `ResultSelector` bidang, hasil tugas ditambahkan di dalam node hasil dalam payload masukan seperti yang ditunjukkan pada contoh berikut:

```
{
  "data": {
```

```
    "firstname": "Jane",
    "lastname": "Doe",
    "identity": {
      "email": "jdoe@example.com",
      "ssn": "123-45-6789"
    },
    "address": {
      "street": "123 Main St",
      "city": "Columbus",
      "state": "OH",
      "zip": "43219"
    },
  "results": {
    "identity": {
      "approved": true
    }
  }
}
```

Menggunakan OutputPath

Anda dapat memilih sebagian dari output negara setelah aplikasi `ResultPath` untuk lolos ke negara berikutnya. Hal ini mengaktifkan Anda untuk memfilter informasi yang tidak diinginkan, dan meneruskan hanya bagian dari JSON yang Anda perlukan.

Pada contoh berikut, `OutputPath` bidang menyimpan output negara di dalam node hasil:`"OutputPath": "$.results"`. Akibatnya, output akhir dari negara, yang dapat Anda lewati ke keadaan berikutnya adalah sebagai berikut:

```
{
    "addressResult": {
      "approved": true,
      "message": "address validation passed"
    },
    "identityResult": {
      "approved": true,
      "message": "identity validation passed"
    }
}
```

Menggunakan fitur konsol untuk memvisualisasikan aliran data input dan output

Anda dapat memvisualisasikan aliran data input dan output antara status dalam alur kerja Anda menggunakan simulator Aliran data konsol Step Functions atau opsi Tampilan lanjutan di halaman Detail Eksekusi.

Tutorial 8: Kesalahan debug di konsol

Saat Anda bekerja dengan Step Functions, Anda mungkin mengalami error runtime yang timbul karena alasan, seperti:

- Jalur JSON yang tidak valid untuk bidang Variabel di negara bagian. `Choice`
- Negara masalah definisi mesin, seperti tidak ada aturan pencocokan didefinisikan untuk `Choice` negara.
- Ekspresi jalur JSON tidak valid saat menerapkan filter untuk memanipulasi input dan output.
- Kegagalan tugas karena pengecualian fungsi Lambda.
- Kesalahan izin IAM.

Dalam tutorial ini, Anda akan belajar tentang debugging beberapa kesalahan ini menggunakan konsol Step Functions. Untuk informasi selengkapnya, lihat Penanganan kesalahan dalam Step Functions.

Topik

- Debugging galat status Choice jalur tidak valid
- Debugging kesalahan ekspresi jalur JSON saat menerapkan filter input dan output

Debugging galat status Choice jalur tidak valid

Bila Anda menentukan jalur JSON yang salah atau tidak dapat diselesaikan di bidang Variabel `Choice` status atau tidak menentukan aturan yang cocok di negara `Choice` bagian, Anda menerima kesalahan saat menjalankan alur kerja Anda.

Untuk menggambarkan kesalahan jalur yang tidak valid, tutorial ini memperkenalkan kesalahan `Choice` status dalam alur kerja Anda. Anda akan menggunakan mesin CreditCardWorkflownegara dan mengedit definisi untuk memperkenalkan kesalahan.

1. Buka konsol Step Functions dan kemudian pilih CreditCardWorkflowstate machine.

2. Pilih Edit untuk mengedit definisi mesin negara. Membuat perubahan disorot dalam kode berikut untuk definisi mesin negara Anda.

```
{
  "Comment": "A description of my state machine",
  "StartAt": "Get credit limit",
  "States": {
    "Get credit limit": {
      ...
      ...
    },
    "Credit applied >= 5000?": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.Payload",
          "NumericLessThan": 5000,
          "Next": "Auto-approve limit"
        },
        {
          "Variable": "$.Payload",
          "NumericGreaterThanEquals": 5000,
          "Next": "Wait for human approval"
        }
        ],
        "Default": "Wait for human approval"
    },
    ...
    ...
  }
}
```

3. Pilih Simpan dan kemudian pilih Simpan pula.
4. Jalankan mesin negara.
5. Pada halaman Detail Eksekusi eksekusi mesin status Anda, lakukan salah satu hal berikut:

   a. Memilih Penyebab pada pesan kesalahan untuk melihat alasan kegagalan eksekusi.

   b. Pilih Tampilkan detail langkah pada pesan kesalahan untuk melihat langkah yang menyebabkan kesalahan.

6. Dalam Input & Keluaran tab dari Rincian langkah bagian, pilih tombol sakelar Tampilan lanjutan untuk melihat jalur transfer data input dan output untuk status yang dipilih.

7. Dalam tampilan Grafik, pastikan Kredit diterapkan >= 5000? dipilih dan lakukan hal berikut:

   a. Lihat nilai masukan negara masuk Masukan kotak.

   b. Pilih tab Definition, dan perhatikan jalur JSON yang ditentukan untuk bidang Variabel.

      Nilai input untuk Kredit yang diterapkan >= 5000? state adalah nilai numerik, sementara Anda telah menentukan jalur JSON untuk nilai input sebagai. `$.Payload` Selama eksekusi mesin negara, `Choice` status tidak dapat menyelesaikan jalur JSON ini karena tidak ada.

8. Edit mesin negara untuk menentukan nilai bidang Variabel sebagai**$**.

```
{
  "Comment": "A description of my state machine",
  "StartAt": "Get credit limit",
  "States": {
    "Get credit limit": {
      ...
      ...
    },
    "Credit applied >= 5000?": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$",
          "NumericLessThan": 5000,
          "Next": "Auto-approve limit"
        },
        {
          "Variable": "$",
          "NumericGreaterThanEquals": 5000,
          "Next": "Wait for human approval"
        }
        ],
        "Default": "Wait for human approval"
    },
    ...
    ...
  }
}
```

Debugging kesalahan ekspresi jalur JSON saat menerapkan filter input dan output

Saat Anda bekerja dengan filter input dan output, Anda mungkin mengalami kesalahan runtime yang timbul karena menentukan ekspresi jalur JSON yang tidak valid.

Contoh berikut menggunakan mesin WorkflowInputOutputnegara yang Anda buat di Tutorial 5 dan menunjukkan skenario di mana Anda menggunakan `ResultSelector` filter untuk memilih bagian dari output tugas.

1. Terapkan `ResultSelector` filter untuk memilih bagian dari output tugas untuk langkah Verifikasi identitas. Untuk melakukan ini, edit definisi mesin negara Anda sebagai berikut:

```
{
  "StartAt": "Verify identity",
  "States": {
    "Verify identity": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "Parameters": {
        "FunctionName": "arn:aws:lambda:us-east-2:123456789012:function:check-
identity",
        "Payload": {
          "email": "jdoe@example.com",
          "ssn": "123-45-6789"
        }
      },
      ...
      ...
      "ResultSelector": {
        "identity.$": "$.Payload.body.message"
 }",
      "End": true
    }
  }
}
```

2. Jalankan mesin negara.
3. Pada halaman Detail Eksekusi eksekusi mesin status Anda, lakukan hal berikut:

   a. Memilih Penyebab pada pesan kesalahan untuk melihat alasan kegagalan eksekusi.

b. Pilih Tampilkan detail langkah pada pesan kesalahan untuk melihat langkah yang menyebabkan kesalahan.

4. Dalam pesan kesalahan, perhatikan bahwa isi simpul $.payload.body adalah string JSON yang lolos. Kesalahan telah terjadi karena Anda tidak dapat merujuk ke string menggunakan notasi jalur JSON.
5. Untuk merujuk ke simpul $.payload.body.Message, lakukan hal berikut:

   a. Gunakan fungsi `States.StringToJSON` intrinsik untuk pertama mengkonversi string ke format JSON.

   b. Tentukan jalur JSON untuk simpul $.payload.body.Message di dalam fungsi intrinsik.

```
"ResultSelector": {
 "identity.$":"States.StringToJson($.Payload.body.message)"
}
```

6. Jalankan mesin negara lagi.

Kasus penggunaan

AWS Step Functions memungkinkan Anda membangun alur kerja visual yang membantu menerjemahkan kebutuhan bisnis dengan cepat ke dalam aplikasi. Step Functions mengelola kondisi, titik pemeriksaan dan memulai ulang untuk Anda, dan menyediakan kemampuan bawaan untuk secara otomatis menangani kesalahan dan pengecualian. Untuk lebih memahami kemampuan yang dapat disediakan Step Functions, baca kasus penggunaan berikut:

Topik

- Pemrosesan data
- Machine Learning
- Orkestrasi mikro
- IT dan otomatisasi keamanan

Pemrosesan data

Seiring berkembangnya volume data yang berasal dari sumber yang semakin beragam, organisasi menyadari mereka perlu bergerak cepat untuk memproses data ini guna memastikan mereka membuat keputusan bisnis yang lebih cepat dan terinformasi dengan baik. Untuk memproses data dalam skala besar, organisasi perlu menyediakan sumber daya secara elastis untuk mengelola informasi yang mereka terima dari perangkat seluler, aplikasi, satelit, pemasaran dan penjualan, penyimpanan data operasional, infrastruktur, dan banyak lainnya.

Step Functions menyediakan skalabilitas, keandalan, dan ketersediaan yang diperlukan untuk berhasil mengelola alur kerja pemrosesan data Anda. Anda dapat mengelola jutaan eksekusi bersamaan dengan Step Functions karena skala horizontal dan menyediakan alur kerja yang menoleransi kesalahan. Memproses data dengan lebih cepat menggunakan eksekusi paralel seperti tipe status `Parallel` Step Functions, atau paralelisme dinamis menggunakan tipe status `Map`. Sebagai bagian dari alur kerja, Anda dapat menggunakan status `Map` untuk mengulangi objek dalam penyimpanan data statis seperti bucket Amazon S3. Step Functions juga memungkinkan Anda dengan mudah kembali mencoba eksekusi yang gagal, atau memilih cara tertentu untuk menangani kesalahan tanpa perlu mengelola proses yang kompleks.

Tergantung pada kebutuhan pemrosesan data Anda, Step Functions langsung terintegrasi dengan layanan pengolahan data lain yang disediakan oleh AWS seperti AWS Batch untuk pemrosesan

batch, Amazon EMR untuk pemrosesan big data, AWS Glue untuk persiapan data, Athena untuk analisis data, dan AWS Lambdauntuk komputasi.

Contoh tipe alur kerja pemrosesan data yang digunakan pelanggan untuk diselesaikan Step Functions meliputi:

Pengolahan file, video, dan gambar

- Ambil koleksi file video dan ubah ke ukuran atau resolusi lain yang ideal untuk perangkat yang akan ditampilkan, seperti ponsel, laptop, atau televisi.
- Ambil sebagian besar koleksi foto yang diunggah oleh pengguna dan ubah menjadi gambar mini atau berbagai citra beresolusi yang kemudian dapat ditampilkan di situs web pengguna.
- Ambil data semi-terstruktur, seperti file CSV, dan gabungkan dengan data yang tidak terstruktur, seperti faktur, untuk menghasilkan laporan bisnis yang dikirim ke pemangku kepentingan bisnis setiap bulan.
- Ambil data mengamati bumi yang dikumpulkan dari satelit, mengubahnya menjadi format yang sejajar satu sama lain dan kemudian tambahkan sumber data lain yang dikumpulkan di bumi untuk wawasan tambahan.
- Ambil log transportasi dari berbagai moda transportasi untuk produk dan mencoba pengoptimalan menggunakan Monte Carlo Simulasi kemudian mengirim laporan kembali ke organisasi dan orang-orang yang mengandalkan Anda untuk mengirimkan barang-barang mereka.

Koordinat ekstrak, transformasi dan beban (ETL) pekerjaan:

- Gabungkan catatan peluang penjualan dengan set data metrik pemasaran melalui serangkaian langkah persiapan data menggunakan AWS Glue, dan menghasilkan kecerdasan bisnis yang dapat digunakan di seluruh organisasi.
- Membuat, memulai, dan mengakhiri klaster Amazon EMR untuk pengolahan big data.

Pemrosesan batch dan beban kerja Komputasi Kinerja Tinggi (HPC):

- Bangun alur analisis sekunder genomik yang memproses seluruh urutan genom mentah ke dalam panggilan varian. Sejajarkan file mentah ke urutan referensi, dan panggil varian pada daftar kromosom tertentu menggunakan paralelisme dinamis.
- Temukan efisiensi dalam produksi perangkat seluler Anda berikutnya atau elektronik lainnya dengan mensimulasikan berbagai tata letak menggunakan senyawa listrik dan kimia yang berbeda.

Jalankan pemrosesan batch besar beban kerja Anda melalui berbagai simulasi untuk mendapatkan desain yang optimal.

Machine Learning

Machine learning memungkinkan organisasi menganalisis data yang dikumpulkan dengan cepat untuk mengidentifikasi pola, kemudian membuat keputusan dengan intervensi manusia yang minimal. Machine learning memulai dengan serangkaian data, diketahui sebagai data pelatihan. Data pelatihan ini membantu meningkatkan akurasi prediksi model machine learning, dan berfungsi sebagai dasar yang digunakan untuk belajar oleh model ini. Setelah model dianggap cukup akurat untuk memenuhi kebutuhan bisnis, model di-deploy ke produksi. AWS Step FunctionsData Science Software Development Kit (SDK) adalah pustaka sumber terbuka yang memungkinkan Anda membuat alur kerja dengan mudah yang memproses data, melatih, dan kemudian mempublikasikan model Anda menggunakan Amazon dan Step Functions. SageMaker

Melakukan praproses serangkaian data yang ada adalah cara yang biasa digunakan organisasi untuk membuat data pelatihan. Metode ini menambahkan informasi, seperti memberi label pada objek dalam citra, teks anotasi atau memproses audio. Untuk memproses data yang dapat Anda gunakanAWS Glue, atau Anda dapat membuat instance SageMaker notebook yang menjalankan aplikasi Jupyter Notebook. Setelah siap, data Anda dapat diunggah ke Amazon S3 untuk akses yang mudah. Selagi model machine learning dilatih, Anda dapat melakukan penyesuaian ke parameter setiap model untuk meningkatkan akurasi sampai model dinyatakan siap untuk deployment.

Step Functions memungkinkan Anda untuk mengatur alur kerja machine learning end-to-end. SageMaker Alur kerja ini dapat mencakup praproses data, pasca-pemrosesan, rekayasa fitur, validasi data, dan evaluasi model. Setelah model di-deploy untuk produksi, Anda dapat memperbaiki dan menguji pendekatan baru untuk terus meningkatkan hasil bisnis. Anda dapat membuat alur kerja siap produksi secara langsung dengan Python, atau Anda dapat menggunakan SDK Ilmu Data Step Functions untuk menyalin alur kerja tersebut, bereksperimen dengan opsi baru, dan menempatkan alur kerja yang disempurnakan dalam produksi.

Beberapa jenis alur kerja machine learning yang digunakan pelanggan untuk Step Functions meliputi:

Deteksi Penipuan

- Identifikasi dan mencegah terjadinya transaksi penipuan, seperti penipuan kredit.
- Deteksi dan cegah pengambilalihan akun menggunakan model machine learning terlatih.

- Identifikasi penyalahgunaan promosi, termasuk pembuatan akun palsu, agar Anda dapat dengan cepat mengambil tindakan.

Personalisasi dan Rekomendasi

- Rekomendasi produk bagi pelanggan yang dituju berdasarkan hal yang diprediksi untuk menarik minat mereka.
- Prediksi kecenderungan pelanggan untuk meningkatkan akun mereka dari tingkat gratis ke langganan berbayar.

Pengayaan Data

- Gunakan pengayaan data sebagai bagian dari praproses untuk memberikan data pelatihan yang lebih baik untuk model machine learning yang lebih akurat.
- Anotasi kutipan teks dan audio untuk menambahkan informasi sintaksis, seperti sarkasme dan bahasa gaul.
- Beri label pada objek tambahan dalam citra untuk memberikan informasi penting sebagai pembelajaran bagi, seperti wujud objek merupakan apel, bola basket, batu, atau binatang.

Orkestrasi mikro

Arsitektur microservice memecah aplikasi menjadi layanan penggabungan longgar. Manfaat dari hal tersebut adalah peningkatan skalabilitas, peningkatan ketahanan, dan waktu untuk memasarkan yang lebih cepat. Setiap microservice independen, sehingga mudah untuk menaikkan skala layanan atau fungsi tunggal tanpa perlu menskalakan seluruh aplikasi. Layanan individu digabungkan secara longgar, memberi kesempatan bagi tim independen untuk fokus pada proses bisnis tunggal, tanpa perlu memahami keseluruhan aplikasi. Microservice juga memungkinkan Anda memilih komponen individual yang sesuai dengan kebutuhan bisnis Anda, memberikan fleksibilitas untuk mengubah pilihan Anda tanpa menulis ulang seluruh alur kerja Anda. Tim yang berbeda dapat menggunakan bahasa pemrograman dan kerangka kerja pilihan mereka untuk bekerja dengan microservice, dan microservice ini masih dapat berkomunikasi dengan yang lain dalam aplikasi melalui Antarmuka Program Aplikasi (API)

Step Functions memberi Anda beberapa cara untuk mengelola alur kerja microservice Anda. Untuk alur kerja yang berjalan lama, Anda dapat menggunakan Alur Kerja Standar dengan AWS Fargate integrasi untuk mengatur aplikasi yang berjalan dalam kontainer. Untuk alur kerja berdurasi

pendek dan bervolume tinggi yang memerlukan respons langsung, Alur Kerja Express Sinkron adalah hal yang ideal. Ini dapat digunakan untuk aplikasi berbasis web atau seluler, yang sering memiliki alur kerja berdurasi pendek, dan memerlukan penyelesaian serangkaian langkah sebelum mengembalikan respons. Anda dapat secara langsung memicu Alur Kerja Express Sinkron dari Amazon API Gateway, dan koneksi diadakan terbuka sampai alur kerja selesai atau batas waktu. Untuk alur kerja berdurasi pendek yang tidak memerlukan respons langsung, Step Functions menyediakan Alur Kerja Express Asinkron.

Contoh beberapa orkestrasi API yang menggunakan Step Functions meliputi:

Alur kerja sinkron atau real-time

- Mengubah nilai dalam catatan seperti memperbarui nama belakang karyawan dan segera tampilkan perubahan di layar.
- Memperbarui pesanan selama checkout, seperti menambahkan, menghapus, atau mengubah kuantitas item, kemudian segera tampilkan pembaruan kembali ke pelanggan.
- Jalankan tugas pemrosesan cepat dan segera kembalikan hasilnya ke peminta.

Orkestrasi Kontainer

- Jalankan tugas di Kubernetes dengan Amazon Elastic Kubernetes Service atau Amazon Elastic Container Service (ECS) dengan Fargate dan integrasikan dengan layanan AWS lainnya, seperti mengirim notifikasi dengan Amazon SNS, sebagai bagian dari alur kerja yang sama.

IT dan otomatisasi keamanan

Otomatisasi IT dapat membantu mengelola operasi yang semakin kompleks dan memakan waktu, seperti meningkatkan dan menambal perangkat lunak, men-deploy pembaruan keamanan untuk mengatasi kelemahan, memilih infrastruktur, menyinkronkan data, merutekan tiket dukungan, dan lainnya. Otomatisasi tugas berulang dan memakan waktu dapat memungkinkan organisasi Anda menyelesaikan operasi rutin dengan cepat dan konsisten dalam skala besar. Hal ini memungkinkan Anda fokus pada pekerjaan strategis seperti pengembangan fitur, permintaan dukungan yang kompleks, dan inovasi selagi memenuhi permintaan yang terus berkembang.

Step Functions memungkinkan Anda membuat alur kerja yang menskalakan secara otomatis untuk memenuhi kebutuhan bisnis Anda tanpa memerlukan intervensi manual. Untuk kasus kesalahan yang terjadi dalam alur kerja Anda, sering kali tidak memerlukan intervensi manual. Step Functions

memungkinkan Anda secara otomatis mencoba kembali tugas yang gagal dan sebuah backoff eksponensial yang dapat mengelola kesalahan dalam alur kerja Anda.

Akan ada situasi yang memerlukan intervensi manusia sebelum alur kerja dapat berkembang. Misalnya, menyetujui peningkatan kredit yang substansial mungkin memerlukan persetujuan manusia. Untuk mengelola ini, Anda dapat menentukan percabangan logika di Step Functions, untuk hanya meminta jumlah yang ditetapkan yang memerlukan persetujuan manusia, selagi semua permintaan lainnya diselesaikan secara otomatis. Dalam kasus yang memerlukan persetujuan manusia , Step Functions memungkinkan Anda menjeda alur kerja pada langkah tertentu, menunggu respons, lalu melanjutkan alur kerja setelah respons diterima.

Beberapa tipe alur kerja otomatisasi yang digunakan pelanggan untuk Step Functions meliputi:

Otomatisasi TI

- Insiden pemulihan otomatis seperti membuka port SSH, ruang disk rendah, atau saat akses publik diberikan ke bucket Amazon S3.
- Mengotomatiskan penyebaran AWS CloudFormation StackSets

Otomatisasi keamanan

- Otomatiskan respons terhadap skenario di mana kunci akses pengguna dan pengguna telah diekspos.
- Respons insiden keamanan pemulihan otomatis menurut dengan tindakan kebijakan yang ditetapkan seperti membatasi tindakan terhadap ARN tertentu atau menerapkan tindakan lainnya.
- Peringatkan karyawan tentang email phishing dalam hitungan detik setelah penerimaan.

Persetujuan Manusia

- Mengotomatisasi pelatihan model machine learning lalu memerlukan persetujuan manual dari model oleh ilmuwan data sebelum kemudian secara otomatis men-deploy atau menolak model berdasarkan respons yang diterima.
- Mengotomatisasi perutean umpan balik pelanggan yang diterima berdasarkan analisis sentimen sehingga mereka yang memiliki sentimen negatif segera meningkat untuk peninjauan manual.

Cara Step Functions bekerja

Bagian ini menjelaskan konsep penting untuk membantu Anda terbiasa dengan AWS Step Functions dan memahami cara kerjanya.

Topik

- Alur Kerja Standar vs Ekspres
- Status
- Mode pemrosesan status peta
- Ambang kegagalan yang ditoleransi untuk status Peta Terdistribusi
- Transisi
- Data Mesin Status
- Pengolahan Input dan Output di Step Functions
- Simulator aliran data
- Mengelola penerapan berkelanjutan dengan versi dan alias
- Eksekusi di Step Functions
- Penanganan kesalahan dalam Step Functions
- Panggil AWS Step Functions dari layanan lain
- Baca Konsistensi di Step Functions
- Penandaan di Step Functions

Alur Kerja Standar vs Ekspres

Saat Anda membuat mesin status, Anda memilih Jenis Standar atau Ekspres. Jenis default untuk mesin negara adalah Standar. Mesin negara yang Tipe Standar disebut alur kerja Standar dan mesin status yang Tipe Ekspres disebut alur kerja Ekspres.

Untuk alur kerja Standar dan Ekspres, Anda menentukan mesin status Anda menggunakan. Amazon States Language Eksekusi mesin status Anda akan berperilaku berbeda tergantung pada Jenis yang Anda pilih.

> **Important**
>
> Jenis yang Anda pilih tidak dapat diubah setelah Anda membuat mesin status.

> **Note**
>
> Jika Anda menentukan mesin status Anda di luar konsol Fungsi Langkah, seperti di editor pilihan Anda, Anda harus menyimpan definisi mesin status Anda dengan ekstensi .asl.json.

Alur Kerja Standar ideal untuk alur kerja yang berjalan lama (hingga satu tahun), tahan lama, dan dapat diaudit. Anda dapat mengambil riwayat eksekusi lengkap menggunakan Step Functions API hingga 90 hari setelah eksekusi selesai. Alur Kerja Standar mengikuti model yang tepat sekali, di mana tugas dan status Anda tidak pernah dijalankan lebih dari sekali, kecuali jika Anda telah menentukan `Retry` perilaku di ASL. Hal ini membuat Alur Kerja Standar cocok untuk mengatur tindakan non-idempoten, seperti memulai klaster EMR Amazon atau memproses pembayaran. Eksekusi Alur Kerja Standar ditagih sesuai dengan jumlah transisi negara yang diproses.

Alur Kerja Ekspres ideal untuk beban kerja volume tinggi dan pengolahan peristiwa seperti penyerapan data IoT, pengolahan dan transformasi data streaming, dan backend aplikasi seluler. Alur Kerja Ekspres bisa berjalan hingga lima menit. Alur Kerja Ekspres menggunakan at-least-oncemodel, di mana eksekusi berpotensi berjalan lebih dari satu kali. Hal ini membuat Alur Kerja Express ideal untuk mengatur tindakan idempoten seperti mengubah data input dan menyimpan melalui tindakan PUT di Amazon DynamoDB. Eksekusi Alur Kerja Ekspres ditagih oleh jumlah eksekusi, durasi eksekusi, dan memori yang dikonsumsi saat eksekusi dijalankan.

Alur Kerja Standar dan Ekspres dapat dimulai secara otomatis sebagai respons terhadap peristiwa seperti permintaan HTTP dari Amazon API Gateway (API yang dikelola sepenuhnya dalam skala besar), Aturan IoT, dan lebih dari 140 sumber peristiwa lainnya di Amazon. EventBridge

> **Tip**
>
> Untuk menerapkan contoh alur kerja Express ke AndaAkun AWS, lihat Modul 7 - API Gateway, Status Paralel, alur kerja Ekspres dari Lokakarya. AWS Step Functions

Untuk informasi tentang pengalaman konsol untuk eksekusi Alur Kerja Standar dan Ekspres, lihat. Eksekusi Alur Kerja Standar dan Ekspres di konsol

Alur Kerja Standar vs Ekspres

	Alur Kerja Standar	Alur Kerja Ekspres: Sinkron dan Tidak Sinkron
Durasi maksimum	Satu tahun	Lima menit
Tingkat mulai eksekusi yang didukung	Untuk informasi tentang kuota yang terkait dengan tingkat mulai eksekusi yang didukung, lihatKuota terkait throttling tindakan API.	Untuk informasi tentang kuota yang terkait dengan tingkat mulai eksekusi yang didukung, lihatKuota terkait throttling tindakan API.
Tingkat transisi status yang didukung	Untuk informasi tentang kuota yang terkait dengan tingkat transisi status yang didukung, lihatKuota terkait throttling status.	Tidak ada batas
Harga	Harga berdasarkan jumlah transisi negara. Transisi status dihitung setiap kali langkah dalam eksekusi Anda selesai.	Harga berdasarkan jumlah eksekusi yang Anda jalankan, durasinya, dan konsumsi memori.
Riwayat eksekusi	Eksekusi dapat dicantumkan dan dijelaskan dengan Step Functions API. Eksekusi dapat di-debug secara visual melalui konsol. Mereka juga dapat diperiksa di CloudWatch Log dengan mengaktifkan logging pada mesin status Anda. Untuk informasi selengkapnya tentang men-debug eksekusi Alur Kerja Standar di konsol,	Riwayat eksekusi tak terbatas, yaitu, sebanyak entri riwayat eksekusi dipertahankan yang dapat Anda hasilkan dalam periode 5 menit. Eksekusi dapat diperiksa di CloudWatch Log atau konsol Step Functions dengan mengaktifkan logging pada mesin status Anda.

	Alur Kerja Standar	Alur Kerja Ekspres: Sinkron dan Tidak Sinkron
	lihat dan. Eksekusi Alur Kerja Standar dan Ekspres di konsol Melihat dan men-debug eksekusi	Untuk informasi selengkapnya tentang men-debug eksekusi Alur Kerja Ekspres di konsol, lihat dan. Eksekusi Alur Kerja Standar dan Ekspres di konsol Melihat dan men-debug eksekusi
Semantik eksekusi	Eksekusi alur kerja yang tepat sekali.	Alur Kerja Ekspres Asinkron: Eksekusi alur kerja. t-least-once Alur Kerja Ekspres Sinkron: Eksekusi t-most-once alur kerja.
Integrasi layanan	Mendukung semua integrasi dan pola layanan.	Mendukung semua integrasi layanan. **Note** Alur Kerja Ekspres tidak mendukung pola integrasi layanan Job-run (`.sync`) atau Callback (`.waitForTaskToken`).
Kegiatan Step Functions	Mendukung kegiatan Step Functions.	Tidak mendukung aktivitas Step Functions.

Alur kerja Ekspres Sinkron dan Tidak Sinkron

Ada dua jenis Alur Kerja Ekspres yang dapat Anda pilih: Asinkron Alur Kerja Ekspres dan Alur Kerja Ekspres Sinkron.

- Alur Kerja Ekspres Asinkron mengembalikan konfirmasi bahwa alur kerja dimulai, tetapi jangan menunggu alur kerja selesai. Untuk mendapatkan hasilnya, Anda harus melakukan polling CloudWatch Log layanan. Anda dapat menggunakan Alur Kerja Ekspres Asinkron ketika Anda tidak memerlukan keluaran respons langsung, seperti layanan pesan atau pemrosesan data yang tidak bergantung pada layanan lain. Anda dapat memulai Alur Kerja Ekspres Asinkron sebagai respons terhadap suatu peristiwa, dengan alur kerja bersarang di Step Functions, atau dengan menggunakan panggilan API. `StartExecution`
- Alur Kerja Ekspres Sinkron memulai alur kerja, tunggu hingga selesai, lalu kembalikan hasilnya. Alur Kerja Ekspres Sinkron dapat digunakan untuk mengatur layanan mikro. Dengan Alur Kerja Synchronous Express, Anda dapat mengembangkan aplikasi tanpa perlu mengembangkan kode tambahan untuk menangani kesalahan, mencoba ulang, atau menjalankan tugas paralel. Anda dapat menjalankan Alur Kerja Synchronous Express yang dipanggil dari Amazon API GatewayAWS Lambda, atau dengan menggunakan panggilan API. `StartSyncExecution`

  > **Note**
  >
  > Jika Anda menjalankan Step Functions Express Workflows secara sinkron dari konsol, `StartSyncExecution` permintaan akan berlalu setelah 60 detik. Untuk menjalankan Alur Kerja Ekspres secara sinkron selama durasi hingga lima menit, buat `StartSyncExecution` permintaan menggunakan AWS SDK atau AWS Command Line Interface (AWS CLI) alih-alih konsol Step Functions.

  Panggilan API eksekusi Express sinkron tidak berkontribusi pada batas kapasitas akun yang ada. Step Functions menyediakan kapasitas sesuai permintaan dan secara otomatis menskalakan dengan beban kerja yang berkelanjutan. Lonjakan beban kerja dapat di-throtling hingga kapasitas tersedia.

Jaminan eksekusi

Alur Kerja Standar	Alur Kerja Ekspres Tidak Sinkron	Alur Kerja Ekspres Sinkron	
Eksekusi alur kerja yang tepat sekali	t-least-onceEksekusi alur kerja	t-most-onceEksekusi alur kerja	
Status eksekusi secara internal tetap ada di antara transisi negara.	Status eksekusi tidak bertahan di antara transisi status.	Status eksekusi tidak bertahan di antara transisi status.	
Secara otomatis mengembalikan respons idempoten saat memulai eksekusi dengan nama yang sama dengan alur kerja yang sedang berjalan. Alur kerja baru tidak dimulai dan pengecualian dilemparkan setelah alur kerja yang sedang berjalan selesai.	Idempotensi tidak dikelola secara otomatis. Memulai beberapa alur kerja dengan nama yang sama menghasilkan eksekusi bersamaan . Dapat mengakiba tkan hilangnya status alur kerja internal jika logika mesin status tidak idempoten.	Idempotensi tidak dikelola secara otomatis. Step Functions menunggu setelah eksekusi dimulai dan mengembalikan hasil state machine setelah selesai. Alur kerja tidak dimulai ulang jika pengecualian terjadi.	
Data riwayat eksekusi dihapus setelah 90 hari. Nama alur kerja dapat digunakan kembali setelah penghapusan data out-of-date eksekusi.	Riwayat eksekusi tidak ditangkap oleh Step Functions . Logging harus diaktifkan melalui Amazon CloudWatch Logs.	Riwayat eksekusi tidak ditangkap oleh Step Functions . Logging harus diaktifkan melalui Amazon CloudWatch Logs.	

Alur Kerja Standar	Alur Kerja Ekspres Tidak Sinkron	Alur Kerja Ekspres Sinkron	
Untuk memenuhi persyaratan kepatuhan, organisasi, atau peraturan, Anda dapat mengurangi periode retensi riwayat eksekusi menjadi 30 hari dengan mengirimkan permintaan kuota. Untuk melakukan ini, gunakan AWS Support Center Console dan buat kasus baru.			

Optimalisasi biaya menggunakan Alur Kerja Ekspres

Step Functions menentukan harga untuk alur kerja Standar dan Ekspres berdasarkan jenis alur kerja yang Anda gunakan untuk membangun mesin status Anda. Untuk mengoptimalkan biaya alur kerja tanpa server Anda, Anda dapat mengikuti salah satu atau kedua rekomendasi berikut:

Topik

- Tip #1: Alur kerja Nesting Express di dalam alur kerja Standar
- Tip #2: Ubah alur kerja Standar menjadi alur kerja Express

Untuk informasi tentang cara memilih jenis alur kerja Standar atau Ekspres memengaruhi penagihan, lihat AWS Step Functions Harga.

Tip #1: Alur kerja Nesting Express di dalam alur kerja Standar

Step Functions menjalankan alur kerja yang memiliki durasi terbatas dan jumlah langkah. Beberapa alur kerja dapat menyelesaikan eksekusi dalam waktu singkat. Orang lain mungkin memerlukan

kombinasi dari high-event-rate alur kerja dan alur kerja yang berjalan lama. Dengan Step Functions, Anda dapat membangun alur kerja yang besar dan kompleks dari beberapa alur kerja yang lebih kecil dan lebih sederhana.

Misalnya, untuk membangun alur kerja pemrosesan pesanan, Anda dapat menyertakan semua tindakan non-idempoten ke dalam alur kerja Standar. Ini dapat mencakup tindakan, seperti menyetujui pesanan melalui interaksi manusia dan memproses pembayaran. Anda kemudian dapat menggabungkan serangkaian tindakan idempoten, seperti mengirim pemberitahuan pembayaran dan memperbarui inventaris produk, dalam alur kerja Express. Anda dapat membuat sarang alur kerja Express ini dalam alur kerja Standar. Dalam contoh ini, alur kerja Standar dikenal sebagai mesin status induk. Alur kerja Express bersarang dikenal sebagai mesin status anak.

Tip #2: Ubah alur kerja Standar menjadi alur kerja Express

Anda dapat mengonversi alur kerja Standar yang ada menjadi alur kerja Express jika memenuhi persyaratan berikut:

- Alur kerja harus menyelesaikan pelaksanaannya dalam waktu lima menit.
- Alur kerja sesuai dengan model at-least-onceeksekusi. Ini berarti bahwa setiap langkah dalam alur kerja dapat berjalan lebih dari satu kali.
- Alur kerja tidak menggunakan pola integrasi `.waitForTaskToken` atau `.sync` layanan.

> **Important**
>
> Alur kerja ekspres menggunakan Amazon CloudWatch Logs untuk merekam riwayat eksekusi. Anda akan dikenakan biaya tambahan saat menggunakan CloudWatch Log.

Untuk mengonversi alur kerja Standar menjadi alur kerja Express menggunakan konsol

1. Buka Konsol Step Functions.
2. Pada halaman mesin Negara, pilih mesin status tipe Standar untuk membukanya.

> **Tip**
>
> Dari daftar tarik-turun Jenis apa pun, pilih Standar untuk memfilter daftar mesin status dan hanya melihat alur kerja Standar.

3. Pilih Salin ke yang baru.

   Workflow Studio terbuka dalam Mode desain menampilkan alur kerja mesin status yang Anda pilih.
4. (Opsional) Perbarui desain alur kerja.
5. Tentukan nama untuk mesin negara Anda. Untuk melakukan ini, pilih ikon edit di sebelah nama mesin status default MyStateMachine. Kemudian, dalam konfigurasi mesin Negara, tentukan nama di kotak Nama mesin Negara.
6. (Opsional) Dalam konfigurasi mesin State, tentukan pengaturan alur kerja lainnya, seperti jenis mesin status dan peran pelaksanaannya.

   Pastikan untuk Type, Anda memilih Express. Simpan semua pilihan default lainnya pada pengaturan mesin State.

   > **Note**
   >
   > Jika Anda mengonversi alur kerja Standar yang sebelumnya ditentukan dalam AWS CDKatauAWS SAM, Anda harus mengubah nilai `Type` dan `Resource` nama.
7. Dalam kotak dialog Konfirmasi pembuatan peran, pilih Konfirmasi untuk melanjutkan.

   Anda juga dapat memilih Lihat pengaturan peran untuk kembali ke konfigurasi mesin Status.

   > **Note**
   >
   > Jika Anda menghapus IAM role yang Step Functions buat, Step Functions tidak dapat membuatnya kembali nanti. Demikian pula, jika Anda mengubah peran (misalnya, dengan menghapus Step Functions dari principal dalam kebijakan IAM), Step Functions tidak dapat memulihkan pengaturan aslinya nanti.

Untuk informasi selengkapnya tentang praktik dan pedoman terbaik saat mengelola pengoptimalan biaya untuk alur kerja, lihat Membangun alur kerja yang hemat biaya. AWS Step Functions

Status

Negara-negara individu dapat membuat keputusan berdasarkan masukan mereka, melakukan tindakan dari input tersebut, dan meneruskan output ke negara lain. DiAWS Step Functions, Anda

menentukan alur kerja Anda di Amazon States Language (ASL). Konsol Step Functions menyediakan representasi grafis dari mesin negara Anda untuk membantu memvisualisasikan logika aplikasi Anda.

> **Note**
>
> Jika Anda menentukan mesin status Anda di luar konsol Step Functions, seperti di editor pilihan Anda, Anda harus menyimpan definisi mesin state Anda dengan ekstensi .asl.json.

Negara adalah elemen dalam mesin negara Anda. Status disebut berdasarkan namanya, bisa berupa string, tetapi harus string yang unik di dalam ruang lingkup seluruh mesin status.

Status dapat melakukan berbagai fungsi di mesin status Anda:

- Lakukan beberapa pekerjaan di mesin status Anda (status Tugas)
- Buat pilihan antara cabang eksekusi (status Pilihan)
- Hentikan eksekusi dengan kegagalan atau keberhasilan (status Gagal atau Sukses)
- Lulus input untuk output, atau menyuntikkan beberapa data tetap ke dalam alur kerja (negara Pass)
- Berikan penundaan untuk jangka waktu tertentu, atau hingga tanggal dan waktu yang ditentukan (keadaan Tunggu)
- Mulai cabang eksekusi paralel (status Pararel)
- Langkah ulang secara dinamis (status Peta)

Berikut ini adalah status contoh bernama `HelloWorld` yang melakukan fungsi AWS Lambda.

```
"HelloWorld": {
  "Type": "Task",
  "Resource": "arn:aws:lambda:us-east-1:123456789012:function:HelloFunction",
  "Next": "AfterHelloWorldState",
  "Comment": "Run the HelloWorld Lambda function"
}
```

Status berbagi banyak fitur umum:

- `Type`Bidang yang menunjukkan jenis keadaan apa itu.
- `Comment`Bidang opsional untuk menyimpan komentar yang dapat dibaca manusia tentang, atau deskripsi, status.

- Setiap status (kecuali `Succeed` atau `Fail`) membutuhkan bidang `Next` atau, sebagai alternatif, dapat menjadi status terminal dengan menentukan bidang `End`.

> **Note**
>
> Status `Choice` dapat memiliki lebih dari satu `Next`, tapi hanya satu dalam Aturan Pilihan. Sebuah `Choice` negara tidak dapat digunakan`End`.

Tipe status tertentu memerlukan bidang tambahan, atau mungkin menentukan ulang penggunaan bidang umum.

Setelah Anda membuat dan menjalankan alur kerja Standar, Anda dapat mengakses informasi tentang setiap status, input dan outputnya, saat aktif dan berapa lama, dengan melihat halaman Detail Eksekusi di konsol Step Functions. Untuk informasi selengkapnya, lihat Melihat dan men-debug eksekusi di konsol Step Functions.

Setelah Anda membuat dan menjalankan eksekusi Alur Kerja Ekspres dan jika pencatatan diaktifkan untuk Alur Kerja Ekspres, Anda dapat mengakses informasi tentang eksekusi di Amazon CloudWatch Logs atau konsol Step Functions. Untuk informasi selengkapnya, lihat Melihat dan men-debug eksekusi di konsol Step Functions.

Topik

- Amazon States Language
- Diteruskan
- Tugas
- Pilihan
- Tunggu
- Berhasil
- Gagal
- Paralel
- Map

Amazon States Language

Amazon States Language adalah, bahasa terstruktur berbasis JSON yang digunakan untuk menentukan mesin status Anda, koleksi status, yang dapat melakukan pekerjaan (status `Task`), menentukan status yang akan ditransisikan ke berikutnya (status `Choice`), menghentikan eksekusi dengan kesalahan (status `Fail`), dan sebagainya.

Untuk informasi selengkapnya, lihat Spesifikasi Amazon States Language dan Statelint, alat yang memvalidasi kode Amazon States Language.

Untuk membuat mesin status pada Konsol Step Functions menggunakan Amazon States Language, lihat Memulai.

> **Note**
>
> Jika Anda menentukan mesin status Anda di luar konsol Step Functions', seperti di editor pilihan Anda, Anda harus menyimpan definisi mesin state Anda dengan ekstensi .asl.json.

Contoh Spesifikasi Amazon States Language

```
{
  "Comment": "An example of the Amazon States Language using a choice state.",
  "StartAt": "FirstState",
  "States": {
    "FirstState": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:FUNCTION_NAME",
      "Next": "ChoiceState"
    },
    "ChoiceState": {
      "Type" : "Choice",
      "Choices": [
        {
          "Variable": "$.foo",
          "NumericEquals": 1,
          "Next": "FirstMatchState"
        },
        {
          "Variable": "$.foo",
          "NumericEquals": 2,
```

```
          "Next": "SecondMatchState"
        }
      ],
      "Default": "DefaultState"
    },

    "FirstMatchState": {
      "Type" : "Task",
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:OnFirstMatch",
      "Next": "NextState"
    },

    "SecondMatchState": {
      "Type" : "Task",
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:OnSecondMatch",
      "Next": "NextState"
    },

    "DefaultState": {
      "Type": "Fail",
      "Error": "DefaultStateError",
      "Cause": "No Matches!"
    },

    "NextState": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:FUNCTION_NAME",
      "End": true
    }
  }
}
```

Topik

- Struktur mesin status
- Fungsi intrinsik
- Bidang Status Umum

Struktur mesin status

Mesin status ditentukan menggunakan teks JSON yang menunjukkan struktur yang berisi bidang-bidang berikut.

`Comment` (Opsional)

Deskripsi tentang mesin status yang dapat dibaca manusia.

`StartAt` (Wajib)

String yang harus sama persis (bersifat peka huruf besar kecil) dengan nama salah satu objek status.

`TimeoutSeconds`(Opsional)

Jumlah detik maksimum eksekusi mesin status dapat berjalan. Jika berjalan lebih lama dari waktu yang ditentukan, eksekusi gagal dengan `States.Timeout` Nama kesalahan.

`Version` (Opsional)

Versi Amazon States Language yang digunakan dalam mesin status (default adalah "1.0").

`States` (Wajib)

Objek yang berisi set status yang dibatasi koma.

Bidang `States` berisi Status.

```
{
    "State1" : {
    },

    "State2" : {
    },
    ...
}
```

Mesin status ditentukan oleh status yang dimilikinya dan hubungan diantaranya.

Berikut adalah contohnya.

```
{
  "Comment": "A Hello World example of the Amazon States Language using a Pass state",
  "StartAt": "HelloWorld",
  "States": {
    "HelloWorld": {
      "Type": "Pass",
      "Result": "Hello World!",
      "End": true
```

```
    }
  }
}
```

Ketika eksekusi mesin status ini diluncurkan, sistem dimulai dengan status yang direferensikan dalam bidang `StartAt` (`"HelloWorld"`). Jika status ini memiliki bidang `"End": true`, eksekusi berhenti dan mengembalikan hasilnya. Jika tidak, sistem mencari bidang `"Next":` dan berlanjut dengan status berikutnya. Proses ini berulang sampi sistem mencapai status terminal (status dengan `"Type": "Succeed"`, `"Type": "Fail"`, atau`"End": true`), atau terjadi kesalahan waktu aktif.

Aturan berikut berlaku untuk status dalam mesin status:

- Status dapat muncul dalam urutan apa pun dalam blok terlampir, tetapi urutan daftarnya tidak memengaruhi urutan status berjalan. Isi status menentukan urutan ini.
- Dalam mesin status, hanya ada satu status yang ditetapkan sebagai status `start`, yang ditetapkan oleh nilai bidang `StartAt` dalam struktur tingkat atas. Status ini adalah salah satu yang dijalankan pertama ketika eksekusi dimulai.
- Setiap status dengan bidang `End` adalah `true` dianggap sebagai status `end` (atau `terminal`). Tergantung pada logika mesin status Anda—misalnya, jika mesin status Anda memiliki beberapa cabang eksekusi—Anda mungkin memiliki lebih dari satu status `end`.
- Jika mesin status Anda hanya terdiri dari satu status, maka dapat menjadi status `start` dan status `end`.

Fungsi intrinsik

Amazon States Language menyediakan beberapa fungsi intrinsik, juga dikenal sebagai intrinsik, yang membantu Anda melakukan operasi pemrosesan data dasar tanpa menggunakan status. `Task` Intrinsik adalah konstruksi yang terlihat mirip dengan fungsi dalam bahasa pemrograman. Mereka dapat digunakan untuk membantu pembangun muatan memproses data yang pergi ke dan dari `Resource` bidang suatu `Task` negara.

Di Amazon States Language, fungsi intrinsik dikelompokkan ke dalam kategori berikut, berdasarkan jenis tugas pemrosesan data yang ingin Anda lakukan:

- Intrinsik untuk array
- Intrinsik untuk pengkodean dan decoding data
- Intrinsik untuk perhitungan hash

- Intrinsik untuk manipulasi data JSON
- Intrinsik untuk operasi Matematika
- Intrinsik untuk operasi String
- Intrinsik untuk pembuatan pengenal unik
- Intrinsik untuk operasi generik

> **Note**
>
> - Untuk menggunakan fungsi intrinsik, Anda harus menentukan `.$` nilai kunci dalam definisi mesin status Anda, seperti yang ditunjukkan pada contoh berikut:
>
> ```
> "KeyId.$": "States.Array($.Id)"
> ```
>
> - Anda dapat membuat sarang hingga 10 fungsi intrinsik dalam bidang dalam alur kerja Anda. Contoh berikut menunjukkan bidang bernama *myArn* yang mencakup sembilan fungsi intrinsik bersarang:
>
> ```
> "myArn.$": "States.Format('{}.{}.{}',
>  States.ArrayGetItem(States.StringSplit(States.ArrayGetItem(States.StringSplit($.ImageRe
>  '/'), 2), '.'), 0),
>  States.ArrayGetItem(States.StringSplit(States.ArrayGetItem(States.StringSplit($.ImageRe
>  '/'), 2), '.'), 1))"
> ```

> **Tip**
>
> Jika Anda menggunakan Step Functions di lingkungan pengembangan lokal, pastikan Anda menggunakan versi 1.12.0 atau lebih tinggi untuk dapat menyertakan semua fungsi intrinsik dalam alur kerja Anda.

Bidang yang mendukung fungsi intrinsik

Tabel berikut menunjukkan bidang yang mendukung fungsi intrinsik untuk setiap status.

Bidang yang mendukung fungsi intrinsik

State	Lulus	Tugas	Pilihan	Tunggu	Berhasil	Gagal	Paralel	Peta
InputPath								
Parameter	✓	✓					✓	✓
ResultSelector		✓					✓	✓
ResultPath								
OutputPath								
Variabel								
<ComparisonOperator>Path								
TimeoutSecondsPath								
HeartbeatSecondsPath								
Kredensial		✓						

Intrinsik untuk array

Gunakan intrinsik berikut untuk melakukan manipulasi array.

States.Array

Fungsi `States.Array` intrinsik membutuhkan nol atau lebih argumen. Interpreter mengembalikan array JSON yang berisi nilai-nilai argumen dalam urutan yang disediakan. Misalnya, diberikan input berikut:

```
{
  "Id": 123456
}
```

Anda dapat menggunakan

```
"BuildId.$": "States.Array($.Id)"
```

Yang akan mengembalikan hasil sebagai berikut:

```
“BuildId”: [123456]
```

States.ArrayPartition

Gunakan fungsi `States.ArrayPartition` intrinsik untuk mempartisi array besar. Anda juga dapat menggunakan intrinsik ini untuk mengiris data dan kemudian mengirim muatan dalam potongan yang lebih kecil.

Fungsi intrinsik ini membutuhkan dua argumen. Argumen pertama adalah array, sedangkan argumen kedua mendefinisikan ukuran potongan. Interpreter memotong array input menjadi beberapa array dari ukuran yang ditentukan oleh ukuran chunk. Panjang potongan array terakhir mungkin kurang dari panjang potongan array sebelumnya jika jumlah item yang tersisa dalam array lebih kecil dari ukuran potongan.

Validasi masukan

- Anda harus menentukan array sebagai nilai masukan untuk argumen pertama fungsi.
- Anda harus menentukan bilangan bulat positif bukan nol untuk argumen kedua yang mewakili nilai ukuran potongan.

  Jika Anda menentukan nilai non-integer untuk argumen kedua, Step Functions akan membulatkannya ke integer terdekat.

- Array input tidak dapat melebihi batas ukuran muatan Step Functions sebesar 256 KB.

Misalnya, diberikan array input berikut:

```
{"inputArray": [1,2,3,4,5,6,7,8,9] }
```

Anda dapat menggunakan `States.ArrayPartition` fungsi untuk membagi array menjadi potongan-potongan dari empat nilai:

```
"inputArray.$": "States.ArrayPartition($.inputArray,4)"
```

Yang akan mengembalikan potongan array berikut:

```
{"inputArray": [ [1,2,3,4], [5,6,7,8], [9]] }
```

Pada contoh sebelumnya, `States.ArrayPartition` fungsi menampilkan tiga array. Dua array pertama masing-masing berisi empat nilai, seperti yang didefinisikan oleh ukuran potongan. Array ketiga berisi nilai yang tersisa dan lebih kecil dari ukuran potongan yang ditentukan.

`States.ArrayContains`

Gunakan fungsi `States.ArrayContains` intrinsik untuk menentukan apakah nilai tertentu hadir dalam array. Misalnya, Anda dapat menggunakan fungsi ini untuk mendeteksi jika ada kesalahan dalam iterasi `Map` status.

Fungsi intrinsik ini membutuhkan dua argumen. Argumen pertama adalah array, sedangkan argumen kedua adalah nilai yang akan dicari dalam array.

Validasi masukan

- Anda harus menentukan array sebagai nilai masukan untuk argumen pertama fungsi ini.
- Anda harus menentukan objek JSON yang valid sebagai argumen kedua.
- Array input tidak dapat melebihi batas ukuran muatan Step Functions sebesar 256 KB.

Misalnya, diberikan array input berikut:

```
{
   "inputArray": [1,2,3,4,5,6,7,8,9],
   "lookingFor": 5
```

```
}
```

Anda dapat menggunakan `States.ArrayContains` fungsi untuk menemukan `lookingFor` nilai dalam`inputArray`:

```
"contains.$": "States.ArrayContains($.inputArray, $.lookingFor)"
```

Karena nilai yang `lookingFor` disimpan dalam termasuk dalam`inputArray`, `States.ArrayContains` mengembalikan hasil sebagai berikut:

```
{"contains": true }
```

`States.ArrayRange`

Gunakan fungsi `States.ArrayRange` intrinsik untuk membuat array baru yang berisi berbagai elemen tertentu. Array baru dapat berisi hingga 1000 elemen.

Fungsi ini membutuhkan tiga argumen. Argumen pertama adalah elemen pertama dari array baru, argumen kedua adalah elemen akhir dari array baru, dan argumen ketiga adalah nilai kenaikan antara unsur-unsur dalam array baru.

Validasi masukan

- Anda harus menentukan nilai integer untuk semua argumen.

  Jika Anda menentukan nilai non-integer untuk salah satu argumen, Step Functions akan membulatkannya ke integer terdekat.
- Anda harus menentukan nilai bukan nol untuk argumen ketiga.
- Array yang baru dihasilkan tidak dapat berisi lebih dari 1000 item.

Misalnya, penggunaan `States.ArrayRange` fungsi berikut akan membuat array dengan nilai pertama 1, nilai akhir 9, dan nilai di antara nilai pertama dan akhir meningkat dua untuk setiap item:

```
"array.$": "States.ArrayRange(1, 9, 2)"
```

Yang akan mengembalikan array berikut:

```
{"array": [1,3,5,7,9] }
```

`States.ArrayGetItem`

Fungsi intrinsik ini mengembalikan nilai indeks tertentu. Fungsi ini membutuhkan dua argumen. Argumen pertama adalah array nilai dan argumen kedua adalah indeks array dari nilai untuk kembali.

Misalnya, gunakan yang berikut `inputArray` dan `index` nilai:

```
{
   "inputArray": [1,2,3,4,5,6,7,8,9],
   "index": 5
}
```

Dari nilai-nilai ini, Anda dapat menggunakan `States.ArrayGetItem` fungsi untuk mengembalikan nilai di `index` posisi 5 dalam array:

```
"item.$": "States.ArrayGetItem($.inputArray, $.index)"
```

Dalam contoh ini, `States.ArrayGetItem` akan mengembalikan hasil sebagai berikut:

```
{ "item": 6 }
```

`States.ArrayLength`

Fungsi `States.ArrayLength` intrinsik mengembalikan panjang array. Ini memiliki satu argumen, array untuk mengembalikan panjang.

Misalnya, diberikan array input berikut:

```
{
   "inputArray": [1,2,3,4,5,6,7,8,9]
}
```

Anda dapat menggunakan `States.ArrayLength` untuk mengembalikan panjang`inputArray`:

```
"length.$": "States.ArrayLength($.inputArray)"
```

Dalam contoh ini, `States.ArrayLength` akan mengembalikan objek JSON berikut yang mewakili panjang array:

```
{ "length": 9 }
```

States.ArrayUnique

Fungsi `States.ArrayUnique` intrinsik menghapus nilai duplikat dari array dan mengembalikan array yang hanya berisi elemen unik. Fungsi ini mengambil array, yang dapat unsorted, sebagai argumen tunggal.

Misalnya, berikut ini `inputArray` berisi serangkaian nilai duplikat:

```
{"inputArray": [1,2,3,3,3,3,3,3,4] }
```

Anda dapat menggunakan `States.ArrayUnique` fungsi sebagai dan menentukan array yang ingin Anda hapus nilai duplikat dari:

```
"array.$": "States.ArrayUnique($.inputArray)"
```

`States.ArrayUnique`Fungsi akan mengembalikan array berikut yang hanya berisi elemen unik, menghapus semua nilai duplikat:

```
{"array": [1,2,3,4] }
```

Intrinsik untuk pengkodean dan decoding data

Gunakan fungsi intrinsik berikut untuk menyandikan atau memecahkan kode data berdasarkan skema pengkodean Base64.

States.Base64Encode

Gunakan fungsi `States.Base64Encode` intrinsik untuk menyandikan data berdasarkan skema pengkodean MIME Base64. Anda dapat menggunakan fungsi ini untuk meneruskan data ke AWS layanan lain tanpa menggunakan AWS Lambda fungsi.

Fungsi ini mengambil string data hingga 10.000 karakter untuk dikodekan sebagai satu-satunya argumen.

Misalnya, pertimbangkan `input` string berikut:

```
{"input": "Data to encode" }
```

Anda dapat menggunakan `States.Base64Encode` fungsi untuk menyandikan `input` string sebagai string MIME Base64:

```
"base64.$": "States.Base64Encode($.input)"
```

`States.Base64Encode`Fungsi mengembalikan data dikodekan berikut sebagai tanggapan:

```
{"base64": "RGF0YSB0byBlbmNvZGU=" }
```

`States.Base64Decode`

Gunakan fungsi `States.Base64Decode` intrinsik untuk memecahkan kode data berdasarkan skema decoding MIME Base64. Anda dapat menggunakan fungsi ini untuk meneruskan data ke AWS layanan lain tanpa menggunakan fungsi Lambda.

Fungsi ini mengambil string data yang dikodekan Base64 hingga 10.000 karakter untuk memecahkan kode sebagai satu-satunya argumen.

Misalnya, diberikan input berikut:

```
{"base64": "RGF0YSB0byBlbmNvZGU=" }
```

Anda dapat menggunakan `States.Base64Decode` fungsi untuk memecahkan kode string base64 ke string yang dapat dibaca manusia:

```
"data.$": "States.Base64Decode($.base64)"
```

Ini `States.Base64Decode function` akan mengembalikan data yang diterjemahkan berikut sebagai tanggapan:

```
{"data": "Decoded data" }
```

Intrinsik untuk perhitungan hash

`States.Hash`

Gunakan fungsi `States.Hash` intrinsik untuk menghitung nilai hash dari input yang diberikan. Anda dapat menggunakan fungsi ini untuk meneruskan data ke AWS layanan lain tanpa menggunakan fungsi Lambda.

Fungsi ini membutuhkan dua argumen. Argumen pertama adalah data yang ingin Anda hitung nilai hash. Argumen kedua adalah algoritma hashing yang digunakan untuk melakukan perhitungan hash. Data yang Anda berikan harus berupa string objek yang berisi 10.000 karakter atau kurang.

Algoritma hashing yang Anda tentukan dapat berupa salah satu algoritma berikut:

- `MD5`
- `SHA-1`
- `SHA-256`
- `SHA-384`
- `SHA-512`

Misalnya, Anda dapat menggunakan fungsi ini untuk menghitung nilai hash `Data` string menggunakan yang ditentukan`Algorithm`:

```
{
   "Data": "input data",
   "Algorithm": "SHA-1"
}
```

Anda dapat menggunakan `States.Hash` fungsi untuk menghitung nilai hash:

```
"output.$": "States.Hash($.Data, $.Algorithm)"
```

`States.Hash`Fungsi mengembalikan nilai hash berikut sebagai tanggapan:

```
{"output": "aaff4a450a104cd177d28d18d7485e8cae074b7" }
```

Intrinsik untuk manipulasi data JSON

Gunakan fungsi-fungsi ini untuk melakukan operasi pengolahan data dasar pada objek JSON.

`States.JsonMerge`

Gunakan fungsi `States.JsonMerge` intrinsik untuk menggabungkan dua objek JSON menjadi satu objek. Fungsi ini membutuhkan tiga argumen. Dua argumen pertama adalah objek JSON

yang ingin Anda gabungkan. Argumen ketiga adalah nilai boolean dari. `false` Nilai boolean ini menentukan apakah mode penggabungan mendalam diaktifkan.

Saat ini, Step Functions hanya mendukung mode penggabungan dangkal; oleh karena itu, Anda harus menentukan nilai boolean sebagai. `false` Dalam mode dangkal, jika kunci yang sama ada di kedua objek JSON, kunci objek terakhir mengganti kunci yang sama di objek pertama. Selain itu, objek yang bersarang di dalam objek JSON tidak digabungkan saat Anda menggunakan penggabungan dangkal.

Misalnya, Anda dapat menggunakan `States.JsonMerge` fungsi untuk menggabungkan objek JSON berikut yang berbagi kunci. a

```
{
   "json1": { "a": {"a1": 1, "a2": 2}, "b": 2 },
   "json2": { "a": {"a3": 1, "a4": 2}, "c": 3 }
}
```

Anda dapat menentukan objek json1 dan json2 sebagai input dalam `States.JsonMerge` fungsi untuk menggabungkannya:

```
"output.$": "States.JsonMerge($.json1, $.json2, false)"
```

`States.JsonMerge`Mengembalikan objek JSON digabungkan berikut sebagai hasilnya. Dalam objek JSON yang digabungkan`output`, kunci `json2` objek a menggantikan kunci `json1` objek. a Juga, objek bersarang di kunci `json1` objek dibuang a karena mode dangkal tidak mendukung penggabungan objek bersarang.

```
{
   "output": {
      "a": {"a3": 1, "a4": 2},
      "b": 2,
      "c": 3
   }
}
```

States.StringToJson

`States.StringToJson`Fungsi ini mengambil jalur referensi ke string JSON yang lolos sebagai satu-satunya argumen.

Interpreter menerapkan parser JSON dan mengembalikan formulir JSON masukan yang diurai. Misalnya, Anda dapat menggunakan fungsi ini untuk menghindari string input berikut:

```
{
 "escapedJsonString": "{\"foo\": \"bar\"}"
}
```

Gunakan `States.StringToJson` fungsi dan tentukan `escapedJsonString` sebagai argumen input:

```
States.StringToJson($.escapedJsonString)
```

`States.StringToJson`Fungsi mengembalikan hasil sebagai berikut:

```
{ "foo": "bar" }
```

States.JsonToString

`States.JsonToString`Fungsi ini hanya membutuhkan satu argumen, yaitu Path yang berisi data JSON untuk kembali sebagai string unescaped. Interpreter mengembalikan string yang berisi teks JSON yang mewakili data yang ditentukan oleh Path. Misalnya, Anda dapat memberikan Jalur JSON berikut yang berisi nilai yang diloloskan:

```
{
  "unescapedJson": {
     "foo": "bar"
  }
}
```

Berikan `States.JsonToString` fungsi dengan data yang terkandung di dalam`unescapedJson`:

```
States.JsonToString($.unescapedJson)
```

`States.JsonToString`Fungsi mengembalikan respon berikut:

```
{\"foo\": \"bar\"}
```

Intrinsik untuk operasi Matematika

Gunakan fungsi-fungsi ini untuk melakukan operasi Matematika.

`States.MathRandom`

Gunakan fungsi `States.MathRandom` intrinsik untuk mengembalikan angka acak antara nomor awal yang ditentukan (inklusif) dan nomor akhir (eksklusif).

Anda dapat menggunakan fungsi ini untuk mendistribusikan tugas tertentu antara dua atau lebih sumber daya.

Fungsi ini membutuhkan tiga argumen. Argumen pertama adalah nomor awal, argumen kedua adalah nomor akhir, dan argumen terakhir mengontrol nilai seed. Argumen nilai benih adalah opsional. Jika Anda menggunakan fungsi ini dengan nilai seed yang sama, ia mengembalikan angka yang identik.

> **Important**
>
> Karena `States.MathRandom` fungsi ini tidak mengembalikan angka acak yang aman secara kriptografis, kami menyarankan Anda untuk tidak menggunakannya untuk aplikasi yang sensitif terhadap keamanan.

Validasi masukan

- Anda harus menentukan nilai integer untuk nomor awal dan argumen nomor akhir.

  Jika Anda menentukan nilai non-integer untuk nomor awal atau argumen nomor akhir, Step Functions akan membulatkannya ke bilangan bulat terdekat.

Misalnya, untuk menghasilkan angka acak antara satu dan 999, Anda dapat menggunakan nilai input berikut:

```
{
   "start": 1,
   "end": 999
}
```

Untuk menghasilkan angka acak, berikan `start` dan `end` nilai ke `States.MathRandom` fungsi:

```
"random.$": "States.MathRandom($.start, $.end)"
```

`States.MathRandom`Fungsi mengembalikan nomor acak berikut sebagai respons:

```
{"random": 456 }
```

`States.MathAdd`

Gunakan fungsi `States.MathAdd` intrinsik untuk mengembalikan jumlah dua angka. Misalnya, Anda dapat menggunakan fungsi ini untuk menambah nilai di dalam loop tanpa menjalankan fungsi Lambda.

Validasi masukan

- Anda harus menentukan nilai integer untuk semua argumen.

  Jika Anda menentukan nilai non-integer untuk salah satu atau kedua argumen, Step Functions akan membulatkannya ke integer terdekat.
- Anda harus menentukan nilai integer dalam kisaran -2147483648 dan 2147483647.

Misalnya, Anda dapat menggunakan nilai berikut untuk mengurangi satu dari 111:

```
{
   "value1": 111,
   "step": -1
}
```

Kemudian, gunakan `States.MathAdd` fungsi yang mendefinisikan `value1` sebagai nilai awal, dan `step` sebagai nilai untuk bertambah dengan: `value1`

```
"value1.$": "States.MathAdd($.value1, $.step)"
```

`States.MathAdd`Fungsi akan mengembalikan nomor berikut sebagai tanggapan:

```
{"value1": 110 }
```

Intrinsik untuk operasi String

`States.StringSplit`

Gunakan fungsi `States.StringSplit` intrinsik untuk membagi string menjadi array nilai. Fungsi ini membutuhkan dua argumen. Argumen pertama adalah string dan argumen kedua adalah karakter pembatas yang akan digunakan fungsi untuk membagi string.

Example - Membagi string input menggunakan karakter pembatas tunggal

Untuk contoh ini, gunakan `States.StringSplit` untuk membagi berikut ini`inputString`, yang berisi serangkaian nilai yang dipisahkan koma:

```
{
    "inputString": "1,2,3,4,5",
    "splitter": ","
}
```

Gunakan `States.StringSplit` fungsi dan definisikan `inputString` sebagai argumen pertama, dan karakter pembatas `splitter` sebagai argumen kedua:

```
"array.$": "States.StringSplit($.inputString, $.splitter)"
```

`States.StringSplit`Fungsi mengembalikan array string berikut sebagai hasilnya:

```
{"array":  ["1","2","3","4","5"] }
```

Example - Membagi string input menggunakan beberapa karakter pembatas

Untuk contoh ini, gunakan `States.StringSplit` untuk membagi berikut ini`inputString`, yang berisi beberapa karakter pembatas:

```
{
  "inputString": "This.is+a,test=string",
  "splitter": ".+,="
}
```

Gunakan `States.StringSplit` fungsi sebagai berikut:

```
{
```

```
  "myStringArray.$": "States.StringSplit($.inputString, $.splitter)"
}
```

`States.StringSplit`Fungsi mengembalikan array string berikut sebagai hasilnya:

```
{"myStringArray": [
  "This",
  "is",
  "a",
  "test",
  "string"
]}
```

Intrinsik untuk pembuatan pengenal unik

`States.UUID`

Gunakan fungsi `States.UUID` intrinsik untuk mengembalikan pengidentifikasi unik universal versi 4 (v4 UUID) yang dihasilkan menggunakan angka acak. Misalnya, Anda dapat menggunakan fungsi ini untuk memanggil AWS layanan atau sumber daya lain yang memerlukan parameter UUID atau menyisipkan item dalam tabel DynamoDB.

`States.UUID`Fungsi ini dipanggil tanpa argumen yang ditentukan:

```
"uuid.$": "States.UUID()"
```

Fungsi mengembalikan UUID yang dihasilkan secara acak, seperti pada contoh berikut:

```
{"uuid": "ca4c1140-dcc1-40cd-ad05-7b4aa23df4a8" }
```

Intrinsik untuk operasi generik

`States.Format`

Gunakan fungsi `States.Format` intrinsik untuk membangun string dari nilai literal dan interpolasi. Fungsi ini membutuhkan satu atau lebih argumen. Nilai argumen pertama harus berupa string, dan mungkin termasuk nol atau lebih contoh dari urutan `{}` karakter. Harus ada banyak argumen yang tersisa dalam doa intrinsik seperti halnya kemunculan. `{}` Interpreter

mengembalikan string didefinisikan dalam argumen pertama dengan masing-masing `{}` diganti dengan nilai argumen yang sesuai secara posisi dalam pemanggilan Intrinsik.

Misalnya, Anda dapat menggunakan masukan berikut dari individu`name`, dan `template` kalimat untuk memasukkan nama mereka ke:

```
{
 "name": "Arnav",
 "template": "Hello, my name is {}."
}
```

Gunakan `States.Format` fungsi dan tentukan `template` string dan string untuk disisipkan sebagai pengganti `{}` karakter:

```
States.Format('Hello, my name is {}.', $.name)
```

or

```
States.Format($.template, $.name)
```

Dengan salah satu input sebelumnya, `States.Format` fungsi mengembalikan string selesai sebagai tanggapan:

```
Hello, my name is Arnav.
```

Karakter yang disimpan dalam fungsi intrinsik

Karakter berikut disimpan untuk fungsi intrinsik, dan harus dihilangkan dengan garis miring terbalik ('\') jika Anda ingin karakter tersebut muncul di Nilai: ' { }, dan \.

Jika karakter \ perlu muncul sebagai bagian dari nilai tanpa berfungsi sebagai karakter pelarian, Anda harus menghindarinya dengan garis miring terbalik. Urutan karakter yang diloloskan berikut digunakan dengan fungsi intrinsik:

- String literal `\'` menunjukkan `'`.
- String literal `\{` menunjukkan `{`.
- String literal `\}` menunjukkan `}`.

- String literal \\ menunjukkan \.

Dalam JSON, garis miring terbalik yang terkandung dalam nilai string literal harus dihilangkan dengan garis miring terbalik lain. Daftar ekuivalen untuk JSON adalah:

- String \\\' yang hilang menunjukkan \'.
- String \\\{ yang hilang menunjukkan \{.
- String \\\} yang hilang menunjukkan \}.
- String \\\\ yang hilang menunjukkan \\.

> **Note**
>
> Jika garis miring terbalik \ escape terbuka ditemukan di string pemanggilan intrinsik, penerjemah akan mengembalikan kesalahan runtime.

Bidang Status Umum

`Type` (Wajib)

Tipe status.

`Next`

Nama status berikutnya yang dijalankan ketika status saat ini selesai. Beberapa tipe status, seperti `Choice`, mengizinkan beberapa status transisi.

Jika keadaan saat ini adalah status terakhir dalam alur kerja Anda, atau status terminal, seperti Berhasil atauGagal, Anda tidak perlu menentukan `Next` bidang.

`End`

Menunjuk status ini sebagai status terminal (mengakhiri eksekusi) jika diatur ke `true`. Jumlah status terminal per mesin status bisa berapa saja. Hanya satu `Next` atau `End` yang dapat digunakan dalam suatu status. Beberapa jenis status, sepertiChoice, atau status terminal, seperti Berhasil danGagal, tidak mendukung atau menggunakan `End` bidang.

`Comment` (Opsional)

Menyediakan deskripsi tentang mesin status yang dapat dibaca manusia.

`InputPath` (Opsional)

Jalur yang memilih sebagian input status yang akan diteruskan ke tugas status untuk pemrosesan. Jika dihilangkan, jalur memiliki nilai $ yang menunjuk seluruh input. Untuk informasi selengkapnya, lihat Pemrosesan Input dan Output.

`OutputPath` (Opsional)

Jalur yang memilih sebagian dari output negara untuk diteruskan ke keadaan berikutnya. Jika dihilangkan, ia memiliki nilai $ yang menunjuk seluruh output. Untuk informasi selengkapnya, lihat Pemrosesan Input dan Output.

Diteruskan

Status `Pass` (`"Type": "Pass"`) meneruskan input dan outputnya, tanpa melakukan pekerjaan. Status `Pass` berguna ketika membangun dan men-debug mesin status.

Anda juga dapat menggunakan `Pass` status untuk mengubah input status JSON menggunakan filter, dan kemudian meneruskan data yang diubah ke status berikutnya dalam alur kerja Anda. Untuk informasi tentang transformasi masukan, lihatInputPath, Parameter dan ResultSelector.

Selain bidang status umum, `Pass` mengizinkan bidang berikut.

`Result` (Opsional)

Mengacu pada output dari tugas virtual yang diteruskan ke negara berikutnya. Jika Anda menyertakan `ResultPath` bidang dalam definisi mesin negara Anda, `Result` ditempatkan seperti yang ditentukan oleh `ResultPath` dan diteruskan ke negara berikutnya.

`ResultPath` (Opsional)

Menentukan di mana untuk menempatkan output (relatif terhadap input) dari tugas virtual yang ditentukan dalam`Result`. Input selanjutnya difilter sebagaimana ditentukan oleh bidang `OutputPath` (jika ada) sebelum digunakan sebagai status output. Untuk informasi selengkapnya, lihat Pemrosesan Input dan Output.

`Parameters` (Opsional)

Menciptakan koleksi pasangan kunci-nilai yang akan diteruskan sebagai masukan. Anda dapat menentukan `Parameters` sebagai nilai statis atau memilih dari input menggunakan jalur. Untuk informasi selengkapnya, lihat InputPath, Parameter dan ResultSelector.

Contoh Status Diteruskan

Berikut adalah contoh status `Pass` yang memasukkan beberapa data tetap ke dalam mesin status, mungkin untuk tujuan pengujian.

```
"No-op": {
  "Type": "Pass",
  "Result": {
    "x-datum": 0.381018,
    "y-datum": 622.2269926397355
  },
  "ResultPath": "$.coords",
  "End": true
}
```

Anggap input untuk status ini adalah sebagai berikut.

```
{
  "georefOf": "Home"
}
```

Kemudian output akan menjadi ini.

```
{
  "georefOf": "Home",
  "coords": {
    "x-datum": 0.381018,
    "y-datum": 622.2269926397355
  }
}
```

Tugas

Status `Task` (`"Type": "Task"`) mewakili satu unit kerja yang dilakukan oleh mesin status. Tugas melakukan pekerjaan dengan menggunakan aktivitas atau AWS Lambda fungsi, dengan mengintegrasikan dengan dukungan lainLayanan AWS, atau dengan menjalankan API pihak ketiga, seperti Stripe.

Bahasa Negara Amazon mewakili tugas dengan menyetel jenis status ke `Task` dan dengan menyediakan tugas dengan Nama Sumber Daya Amazon (ARN) aktivitas, fungsi Lambda, atau

titik akhir API pihak ketiga. Definisi status Tugas berikut memanggil fungsi Lambda bernama. *HelloFunction*

```
"Lambda Invoke": {
  "Type": "Task",
  "Resource": "arn:aws:states:::lambda:invoke",
  "Parameters": {
    "Payload.$": "$",
    "FunctionName": "arn:aws:lambda:us-east-2:123456789012:function:HelloFunction:
$LATEST"
  },
  "End": true
}
```

Dalam topik ini:

- Tipe tugas
- Bidang status tugas
- Contoh definisi status tugas
- Aktivitas

Tipe tugas

Step Functions mendukung jenis tugas berikut yang dapat Anda tentukan dalam definisi status Tugas:

- Aktivitas
- Fungsi Lambda
- A didukung Layanan AWS
- Tugas HTTP

Anda menentukan jenis tugas dengan memberikan ARN di `Resource` bidang definisi status Tugas. Contoh berikut menunjukkan sintaks `Resource` bidang. Semua jenis Tugas kecuali yang memanggil API pihak ketiga, gunakan sintaks berikut. Untuk informasi tentang sintaks Tugas HTTP, lihatPanggil API pihak ketiga.

Dalam definisi status Tugas Anda, ganti teks yang dicetak miring dalam sintaks berikut dengan informasi khusus sumber daya. AWS

```
arn:partition:service:region:account:task_type:name
```

Daftar berikut menjelaskan komponen individual dalam sintaks ini:

- `partition` adalah partisi AWS Step Functions untuk digunakan, yang paling sering adalah `aws`.
- `service`menunjukkan yang Layanan AWS digunakan untuk menjalankan tugas, dan dapat menjadi salah satu dari nilai-nilai berikut:
  - `states` untuk aktivitas.
  - `lambda` untuk Fungsi Lambda. Jika Anda mengintegrasikan dengan yang lainLayanan AWS, misalnya, Amazon SNS atau Amazon DynamoDB, gunakan atau. `sns` `dynamodb`
- `region`adalah kode AWS Wilayah di mana aktivitas Step Functions atau jenis mesin status, fungsi Lambda, atau AWS sumber daya lainnya telah dibuat.
- `account`adalah Akun AWS ID di mana Anda telah mendefinisikan sumber daya.
- `task_type` adalah tipe tugas untuk dijalankan. Ini bisa menjadi salah satu nilai berikut:
  - `activity` – Aktivitas.
  - `function` – Fungsi Lambda.
  - *`servicename`* – Nama layanan terhubung yang didukung (lihat Integrasi yang dioptimalkan untuk Step Functions).
- `name` adalah nama sumber daya terdaftar (nama aktivitas, nama fungsi Lambda, atau tindakan API layanan).

> **Note**
>
> Step Functions tidak mendukung referensi ARN di seluruh partisi atau wilayah. Misalnya, tidak `aws-cn` dapat memanggil tugas di `aws` partisi, dan sebaliknya.

Bagian berikut menyediakan detail selengkapnya tentang setiap tipe tugas.

Activity

Aktivitas menunjukkan pekerja (proses atau thread), diimplementasikan dan di-hosting oleh Anda, yang melakukan tugas tertentu. Aktivitas hanya didukung oleh Alur Kerja standar, tidak Alur Kerja Ekspres.

Aktivitas ARN `Resource` menggunakan sintaksis berikut.

```
arn:partition:states:region:account:activity:name
```

> **Note**
>
> Anda harus membuat aktivitas dengan Step Functions (menggunakan aksi API CreateActivity, atau konsol Step Functions) sebelum digunakan pertama kali.

Untuk informasi selengkapnya tentang cara membuat aktivitas dan mengimplementasikan pekerja, lihat Aktivitas.

Fungsi Lambda

Tugas Lambda mengeksekusi fungsi menggunakan AWS Lambda. Untuk menentukan fungsi Lambda, gunakan ARN dari fungsi Lambda di bidang `Resource`.

Bergantung pada jenis integrasi (Integrasi yang dioptimalkan atau integrasi AWS SDK) yang Anda gunakan untuk menentukan fungsi Lambda, sintaks bidang fungsi Lambda Anda bervariasi. `Resource`

Sintaks `Resource` bidang berikut adalah contoh integrasi yang dioptimalkan dengan fungsi Lambda.

```
"arn:aws:states:::lambda:invoke"
```

Sintaks `Resource` bidang berikut adalah contoh integrasi AWS SDK dengan fungsi Lambda.

```
"arn:aws:states:::aws-sdk:lambda:invoke"
```

Definisi `Task` status berikut menunjukkan contoh integrasi yang dioptimalkan dengan fungsi Lambda bernama. *HelloWorld*

```
"LambdaState": {
  "Type": "Task",
  "Resource": "arn:aws:states:::lambda:invoke",
  "OutputPath": "$.Payload",
  "Parameters": {
    "Payload.$": "$",
```

```
    "FunctionName": "arn:aws:lambda:us-east-1:function:HelloWorld:$LATEST"
  },
  "Next": "NextState"
}
```

Setelah fungsi Lambda yang ditentukan di `Resource` bidang selesai, outputnya dikirim ke status yang diidentifikasi di `Next` bidang (" "). NextState

A didukung Layanan AWS

Ketika Anda mereferensikan sumber daya yang terhubung, Step Functions langsung memanggil tindakan API layanan yang didukung. Tentukan layanan dan tindakan di bidang `Resource`.

ARN `Resource` layanan yang terhubung menggunakan sintaksis berikut.

```
arn:partition:states:region:account:servicename:APIname
```

> **Note**
>
> Untuk membuat koneksi sinkron ke sumber daya yang terhubung, tambahkan `.sync` ke entri *APIname* di ARN. Untuk informasi selengkapnya, lihat Bekerja dengan layanan yang lain.

Misalnya:

```
{
 "StartAt": "BATCH_JOB",
 "States": {
   "BATCH_JOB": {
     "Type": "Task",
     "Resource": "arn:aws:states:::batch:submitJob.sync",
     "Parameters": {
       "JobDefinition": "preprocessing",
       "JobName": "PreprocessingBatchJob",
       "JobQueue": "SecondaryQueue",
       "Parameters.$": "$.batchjob.parameters",
       "RetryStrategy": {
          "attempts": 5
        }
     },
     "End": true
```

```
    }
  }
}
```

Bidang status tugas

Selain bidang status umum, status `Task` mempunyai bidang berikut.

`Resource` (Wajib)

URI, terutama ARN yang secara unik mengidentifikasi tugas tertentu untuk mengeksekusi.

`Parameters` (Opsional)

Digunakan untuk menyampaikan informasi ke tindakan API dari sumber daya yang terhubung. Parameter dapat menggunakan campuran JSON statis dan JsonPath. Untuk informasi selengkapnya, lihat Meneruskan parameter ke API layanan.

`Credentials` (Opsional)

Menentukan peran target peran eksekusi mesin negara harus mengasumsikan sebelum memanggil yang ditentukan. `Resource` Atau, Anda juga dapat menentukan nilai JsonPath atau fungsi intrinsik yang menyelesaikan ARN peran IAM saat runtime berdasarkan input eksekusi. Jika Anda menentukan nilai JsonPath, Anda harus awalan dengan notasi. `$.`

Untuk contoh menggunakan bidang ini di `Task` negara bagian, lihatContoh bidang Kredensial status tugas. Untuk contoh menggunakan bidang ini untuk mengakses AWS sumber daya lintas akun dari mesin status Anda, lihatTutorial: Mengakses sumber daya lintas akun AWS.

> **Note**
>
> Bidang ini didukung oleh Tipe tugas yang menggunakan fungsi Lambda dan layanan yang didukung AWS.

`ResultPath` (Opsional)

Menentukan tempat (dalam input) untuk meletakkan hasil eksekusi tugas yang ditentukan di `Resource`. Input kemudian difilter sebagaimana ditentukan oleh bidang `OutputPath` (jika ada) sebelum digunakan sebagai output status. Untuk informasi selengkapnya, lihat Pemrosesan Input dan Output.

`ResultSelector` (Opsional)

Teruskan koleksi pasangan nilai kunci, tempat nilai-nilai bersifat statis atau dipilih dari hasil. Untuk informasi selengkapnya, lihat ResultSelector.

`Retry` (Opsional)

Array objek yang disebut Retriers, yang menentukan kebijakan percobaan ulang jika status menemukan kesalahan waktu aktif. Untuk informasi selengkapnya, lihat Nyatakan contoh mesin menggunakan Coba Ulang dan menggunakan Catch.

`Catch` (Opsional)

Array objek, disebut Catch yang menentukan status fallback. Status ini dijalankan jika status menemukan kesalahan waktu aktif dan kebijakan percobaan ulang habis atau tidak ditentukan. Untuk informasi selengkapnya, lihat Status Fallback.

`TimeoutSeconds` (Opsional)

Menentukan waktu maksimum aktivitas atau tugas dapat berjalan sebelum waktu habis dengan States.Timeout kesalahan dan gagal. Nilai batas waktu harus positif, bilangan bulat bukan nol. Nilai bawaannya adalah 99999999.

Hitungan batas waktu dimulai setelah tugas dimulai, misalnya, kapan `ActivityStarted` atau `LambdaFunctionStarted` peristiwa dicatat dalam riwayat peristiwa eksekusi. Untuk Aktivitas, penghitungan dimulai saat `GetActivityTask` menerima token dan `ActivityStarted` dicatat dalam riwayat peristiwa eksekusi.

Saat tugas dimulai, Step Functions menunggu respons sukses atau gagal dari tugas atau pekerja aktivitas dalam `TimeoutSeconds` durasi yang ditentukan. Jika pekerja tugas atau aktivitas gagal merespons dalam waktu ini, Step Functions menandai eksekusi alur kerja sebagai gagal.

`TimeoutSecondsPath` (Opsional)

Jika Anda ingin memberikan nilai waktu habis dinamis dari input status menggunakan jalur referensi, gunakan `TimeoutSecondsPath`. Ketika diselesaikan, jalur referensi harus memilih bidang yang nilainya adalah bilangan bulat positif.

> **Note**
>
> Suatu `Task` negara tidak dapat mencakup keduanya `TimeoutSeconds` dan`TimeoutSecondsPath`.

`HeartbeatSeconds` (Opsional)

Menentukan frekuensi sinyal detak jantung yang dikirim pekerja aktivitas selama pelaksanaan tugas. Detak jantung menunjukkan bahwa tugas masih berjalan dan perlu lebih banyak waktu untuk menyelesaikannya. Detak jantung mencegah aktivitas atau tugas dari waktu habis dalam durasi. `TimeoutSeconds`

`HeartbeatSeconds`harus berupa nilai bilangan bulat positif, bukan nol kurang dari nilai `TimeoutSeconds` bidang. Nilai bawaannya adalah 99999999. Jika lebih banyak waktu dari detik yang ditentukan berlalu antara detak jantung dari tugas, status Tugas gagal dengan kesalahan. States.Timeout

Untuk Aktivitas, penghitungan dimulai saat `GetActivityTask` menerima token dan `ActivityStarted` dicatat dalam riwayat peristiwa eksekusi.

`HeartbeatSecondsPath` (Opsional)

Jika Anda ingin memberikan nilai heartbeat secara dinamis dari input status menggunakan jalur referensi, gunakan `HeartbeatSecondsPath`. Ketika diselesaikan, jalur referensi harus memilih bidang yang nilainya adalah bilangan bulat positif.

> **Note**
>
> Suatu `Task` negara tidak dapat mencakup keduanya `HeartbeatSeconds` dan`HeartbeatSecondsPath`.

Status `Task` harus mengatur salah satu bidang `End` ke `true` jika status mengakhiri eksekusi, atau harus memberikan status di bidang `Next` yang dijalankan ketika status `Task` selesai.

Contoh definisi status tugas

Contoh berikut menunjukkan bagaimana Anda dapat menentukan definisi status Tugas berdasarkan kebutuhan Anda.

- Menentukan batas waktu status Tugas dan interval detak jantung
  - Contoh notifikasi waktu habis dan heartbeat statis
  - Contoh notifikasi waktu habis dan heartbeat tugas dinamis
- Menggunakan bidang Credentials
  - Menentukan peran IAM hard-code ARN

- Menentukan JSONPath sebagai peran IAM ARN
- Menentukan fungsi intrinsik sebagai peran IAM ARN

Interval waktu habis dan heartbeat status tugas

Ini adalah praktik terbaik untuk mengatur nilai waktu habis dan interval heartbeat untuk aktivitas yang berjalan lama. Hal ini dapat dilakukan dengan menentukan nilai batas waktu dan heartbeat, atau dengan mengaturnya secara dinamis.

Contoh notifikasi waktu habis dan heartbeat statis

Saat `HelloWorld` selesai, status berikutnya (di sini disebut `NextState`) akan dijalankan.

Jika tugas ini gagal diselesaikan dalam waktu 300 detik, atau tidak mengirim notifikasi heartbeat dalam interval 60 detik, tugas ditandai sebagai `failed`.

```
"ActivityState": {
  "Type": "Task",
  "Resource": "arn:aws:states:us-east-1:123456789012:activity:HelloWorld",
  "TimeoutSeconds": 300,
  "HeartbeatSeconds": 60,
  "Next": "NextState"
}
```

Contoh notifikasi waktu habis dan heartbeat tugas dinamis

Dalam contoh ini, saat tugas AWS Glue selesai, status berikutnya akan dijalankan.

Jika tugas ini gagal diselesaikan dalam interval yang ditetapkan secara dinamis oleh AWS Glue pekerjaan, tugas ditandai sebagai`failed`.

```
"GlueJobTask": {
  "Type": "Task",
  "Resource": "arn:aws:states:::glue:startJobRun.sync",
  "Parameters": {
    "JobName": "myGlueJob"
  },
  "TimeoutSecondsPath": "$.params.maxTime",

  "Next": "NextState"
}
```

Contoh bidang Kredensial status tugas

Menentukan peran IAM hard-code ARN

Contoh berikut menentukan peran IAM target yang harus diasumsikan oleh peran eksekusi mesin status untuk mengakses fungsi Lambda lintas akun bernama. `Echo` Dalam contoh ini, peran target ARN ditentukan sebagai nilai hard-code.

```
{
  "StartAt": "Cross-account call",
  "States": {
    "Cross-account call": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "Credentials": {
        "RoleArn": "arn:aws:iam::111122223333:role/LambdaRole"
      },
      "Parameters": {
        "FunctionName": "arn:aws:lambda:us-east-2:111122223333:function:Echo"
      },
      "End": true
    }
  }
}
```

Menentukan JSONPath sebagai peran IAM ARN

Contoh berikut menentukan nilai JsonPath, yang akan menyelesaikan peran IAM ARN saat runtime.

```
{
  "StartAt": "Lambda",
  "States": {
    "Lambda": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "Credentials": {
        "RoleArn.$": "$.roleArn"
      },
      ...
    }
  }
}
```

Menentukan fungsi intrinsik sebagai peran IAM ARN

Contoh berikut menggunakan fungsi `States.Format`intrinsik, yang menyelesaikan peran IAM ARN saat runtime.

```
{
  "StartAt": "Lambda",
  "States": {
    "Lambda": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "Credentials": {
        "RoleArn.$": "States.Format('arn:aws:iam::{}:role/ROLENAME', $.accountId)"
      },
      ...
    }
  }
}
```

Aktivitas

Aktivitas adalah AWS Step Functions fitur yang memungkinkan Anda memiliki tugas di mesin status tempat pekerjaan dilakukan oleh pekerja yang dapat dihosting di Amazon Elastic Compute Cloud (Amazon EC2), Amazon Elastic Container Service (Amazon ECS), perangkat seluler—pada dasarnya di mana saja.

Gambaran Umum

Di AWS Step Functions, aktivitas adalah cara untuk mengaitkan kode yang berjalan di suatu tempat (dikenal sebagai pekerja aktivitas) dengan tugas tertentu dalam mesin status. Anda dapat membuat aktivitas menggunakan konsol Step Functions, atau dengan memanggil `CreateActivity`. Ini menyediakan Amazon Resource Name (ARN) untuk status tugas Anda. Gunakan ARN ini untuk melakukan poll pada status tugas untuk bekerja di pekerja aktivitas Anda.

> **Note**
>
> Aktivitas tidak memiliki versi dan diharapkan kompatibel dengan versi yang lebih lama. Jika Anda harus membuat perubahan yang tidak kompatibel dengan versi yang lebih lama, buat aktivitas baru di Step Functions menggunakan nama yang unik.

Seorang pekerja aktivitas dapat menjadi aplikasi yang berjalan di instans Amazon EC2, fungsi AWS Lambda, perangkat seluler: aplikasi apa pun yang dapat membuat koneksi HTTP, yang di-hosting di mana saja. Ketika Step Functions mencapai status tugas aktivitas, alur kerja menunggu agar pekerja aktivitas melakukan polling untuk tugas. Pekerja aktivitas melakukan polling pada Step Functions dengan menggunakan `GetActivityTask`, dan mengirim ARN untuk aktivitas terkait. `GetActivityTask` mengembalikan respons termasuk `input` (string input JSON untuk tugas) dan `taskToken` (pengidentifikasi unik untuk tugas). Setelah pekerja aktivitas menyelesaikan pekerjaannya, laporan keberhasilan atau kegagalan dapat diberikan menggunakan `SendTaskSuccess` atau `SendTaskFailure`. Kedua panggilan ini menggunakan `taskToken` yang disediakan oleh `GetActivityTask` untuk mengaitkan hasilnya dengan tugas tersebut.

API Terkait dengan Tugas Aktivitas

Step Functions menyediakan API untuk membuat dan mendaftar aktivitas, meminta tugas, dan untuk mengelola alur mesin status Anda berdasarkan hasil pekerja Anda.

Berikut ini adalah API Step Functions yang terkait dengan aktivitas:

- `CreateActivity`
- `GetActivityTask`
- `ListActivities`
- `SendTaskFailure`
- `SendTaskHeartbeat`
- `SendTaskSuccess`

> **Note**
>
> Polling untuk tugas aktivitas dengan `GetActivityTask` dapat menyebabkan latensi dalam beberapa implementasi. Lihat Hindari latensi saat polling untuk tugas aktivitas.

Menunggu tugas aktivitas Selesai

Konfigurasikan berapa lama status menunggu dengan mengatur `TimeoutSeconds` dalam ketentuan tugas. Agar tugas tetap aktif dan menunggu, kirim heartbeat secara berkala dari pekerja aktivitas Anda menggunakan `SendTaskHeartbeat` dalam waktu yang dikonfigurasi di `TimeoutSeconds`.

Dengan mengonfigurasi durasi waktu habis yang panjang dan mengirim heartbeat secara aktif, aktivitas di Step Functions dapat menunggu hingga satu tahun untuk menyelesaikan eksekusi.

Misalnya, jika Anda memerlukan alur kerja yang menunggu outcome proses yang panjang, lakukan hal berikut:

1. Buat aktivitas menggunakan konsol tersebut, atau dengan menggunakan `CreateActivity`. Perhatikan ARN aktivitas.
2. Referensikan bahwa ARN di status tugas aktivitas dalam ketentuan mesin status dan atur `TimeoutSeconds`.
3. Implementasikan pekerja aktivitas yang melakukan polling untuk bekerja menggunakan `GetActivityTask`, yang mereferensikan ARN aktivitas tersebut.
4. Gunakan `SendTaskHeartbeat` secara berkala dalam waktu yang Anda tetapkan di `HeartbeatSeconds` dalam ketentuan tugas mesin status Anda untuk menjaga tugas dari waktu habis.
5. Mulai eksekusi mesin status Anda.
6. Mulai proses pekerja aktivitas Anda.

Eksekusi berhenti di status tugas aktivitas dan menunggu pekerja aktivitas Anda untuk melakukan polling untuk tugas. Setelah `taskToken` disediakan untuk pekerja aktivitas Anda, alur kerja Anda akan menunggu untuk `SendTaskSuccess` atau `SendTaskFailure` untuk memberikan status. Jika eksekusi tidak menerima salah satu dari hal ini atau panggilan `SendTaskHeartbeat` sebelum waktu dikonfigurasi dalam `TimeoutSeconds`, eksekusi akan gagal dan riwayat eksekusi akan berisi peristiwa `ExecutionTimedOut`.

Langkah Selanjutnya

Untuk tampilan yang lebih detail dalam pembuatan mesin status yang menggunakan pekerja aktivitas, lihat:

- Membuat mesin status Aktivitas menggunakan Step Functions
- Contoh Activity Worker di Ruby

Contoh Activity Worker di Ruby

Berikut ini adalah contoh aktivitas pekerja yang menggunakan AWS SDK for Ruby untuk menunjukkan cara menggunakan praktik terbaik dan mengimplementasikan pekerja aktivitas Anda sendiri.

Kode menerapkan pola konsumen-produsen dengan jumlah thread yang dikonfigurasi untuk poller dan pekerja aktivitas. Thread poller terus-menerus melakukan polling tugas aktivitas. Setelah tugas aktivitas diambil, tugas diteruskan melalui antrean blok yang dibatasi untuk diambil thread aktivitas.

- Untuk informasi selengkapnya tentang AWS SDK for Ruby, lihat Referensi API AWS SDK for Ruby.
- Untuk mengunduh kode ini dan sumber daya terkait, lihat repositori step-functions-ruby-activity-worker aktif. GitHub

Kode Ruby berikut adalah entri point utama untuk contoh pekerja aktivitas Ruby ini.

```
require_relative '../lib/step_functions/activity'
credentials = Aws::SharedCredentials.new
region = 'us-west-2'
activity_arn = 'ACTIVITY_ARN'

activity = StepFunctions::Activity.new(
  credentials: credentials,
  region: region,
  activity_arn: activity_arn,
  workers_count: 1,
  pollers_count: 1,
  heartbeat_delay: 30
)

# The start method takes as argument the block that is the actual logic of your custom
activity.
activity.start do |input|
  { result: :SUCCESS, echo: input['value'] }
end
```

Kode menyertakan default yang dapat Anda ubah untuk mereferensikan aktivitas Anda, dan untuk menyesuaikannya dengan implementasi tertentu Anda. Kode ini mengambil logika implementasi aktual sebagai input, memungkinkan Anda untuk mereferensikan aktivitas dan kredensial spesifik Anda, dan memungkinkan Anda untuk mengonfigurasi jumlah thread dan penundaan heartbeat.

Untuk informasi selengkapnya dan untuk mengunduh kode, lihat Pekerja Aktivitas Ruby Step Functions.

Item	Deskripsi
`require_relative`	Jalur relatif ke kode pekerja aktivitas contoh berikut.
`region`	Wilayah AWS aktivitas Anda.
`workers_count`	Jumlah thread untuk pekerja aktivitas Anda. 10 hingga 20 thread harusnya sudah cukup untuk sebagian besar implementasi. Semakin lama waktu yang dibutuhkan aktivitas yang untuk memproses, semakin banyak thread yang dibutuhkan. Sebagai perkiraan, kalikan jumlah aktivitas proses per detik dengan latensi pemrosesan aktivitas persentil ke-99, dalam detik.
`pollers_count`	Jumlah thread untuk poller Anda. 10 hingga 20 thread harusnya sudah cukup untuk sebagian besar implementasi.
`heartbeat_delay`	Penundaan dalam detik antara heartbeat.
`input`	Logika implementasi aktivitas Anda.

Berikut ini adalah pekerja aktivitas Ruby, yang direferensikan dengan `../lib/step_functions/activity` dalam kode Anda.

```
require 'set'
require 'json'
require 'thread'
require 'logger'
require 'aws-sdk'

module Validate
```

```
  def self.positive(value)
    raise ArgumentError, 'Argument has to be positive' if value <= 0
    value
  end

  def self.required(value)
    raise ArgumentError, 'Argument is required' if value.nil?
    value
  end
end

module StepFunctions
  class RetryError < StandardError
    def initialize(message)
      super(message)
    end
  end

  def self.with_retries(options = {}, &block)
    retries = 0
    base_delay_seconds = options[:base_delay_seconds] || 2
    max_retries = options[:max_retries] || 3
    begin
      block.call
    rescue => e
      puts e
      if retries < max_retries
        retries += 1
        sleep base_delay_seconds**retries
        retry
      end
      raise RetryError, 'All retries of operation had failed'
    end
  end

  class Activity
    def initialize(options = {})
      @states = Aws::States::Client.new(
        credentials: Validate.required(options[:credentials]),
        region: Validate.required(options[:region]),
        http_read_timeout: Validate.positive(options[:http_read_timeout] || 60)
      )
      @activity_arn = Validate.required(options[:activity_arn])
      @heartbeat_delay = Validate.positive(options[:heartbeat_delay] || 60)
```

```
      @queue_max = Validate.positive(options[:queue_max] || 5)
      @pollers_count = Validate.positive(options[:pollers_count] || 1)
      @workers_count = Validate.positive(options[:workers_count] || 1)
      @max_retry = Validate.positive(options[:workers_count] || 3)
      @logger = Logger.new(STDOUT)
    end

    def start(&block)
      @sink = SizedQueue.new(@queue_max)
      @activities = Set.new
      start_heartbeat_worker(@activities)
      start_workers(@activities, block, @sink)
      start_pollers(@activities, @sink)
      wait
    end

    def queue_size
      return 0 if @sink.nil?
      @sink.size
    end

    def activities_count
      return 0 if @activities.nil?
      @activities.size
    end

    private

    def start_pollers(activities, sink)
      @pollers = Array.new(@pollers_count) do
        PollerWorker.new(
          states: @states,
          activity_arn: @activity_arn,
          sink: sink,
          activities: activities,
          max_retry: @max_retry
        )
      end
      @pollers.each(&:start)
    end

    def start_workers(activities, block, sink)
      @workers = Array.new(@workers_count) do
        ActivityWorker.new(
```

```
          states: @states,
          block: block,
          sink: sink,
          activities: activities,
          max_retry: @max_retry
        )
      end
      @workers.each(&:start)
    end

    def start_heartbeat_worker(activities)
      @heartbeat_worker = HeartbeatWorker.new(
        states: @states,
        activities: activities,
        heartbeat_delay: @heartbeat_delay,
        max_retry: @max_retry
      )
      @heartbeat_worker.start
    end

    def wait
      sleep
    rescue Interrupt
      shutdown
    ensure
      Thread.current.exit
    end

    def shutdown
      stop_workers(@pollers)
      wait_workers(@pollers)
      wait_activities_drained
      stop_workers(@workers)
      wait_activities_completed
      shutdown_workers(@workers)
      shutdown_worker(@heartbeat_worker)
    end

    def shutdown_workers(workers)
      workers.each do |worker|
        shutdown_worker(worker)
      end
    end
```

```
    def shutdown_worker(worker)
      worker.kill
    end

    def wait_workers(workers)
      workers.each(&:wait)
    end

    def wait_activities_drained
      wait_condition { @sink.empty? }
    end

    def wait_activities_completed
      wait_condition { @activities.empty? }
    end

    def wait_condition(&block)
      loop do
        break if block.call
        sleep(1)
      end
    end

    def stop_workers(workers)
      workers.each(&:stop)
    end

    class Worker
      def initialize
        @logger = Logger.new(STDOUT)
        @running = false
      end

      def run
        raise 'Method run hasn\'t been implemented'
      end

      def process
        loop do
          begin
            break unless @running
            run
          rescue => e
            puts e
```

```
            @logger.error('Unexpected error has occurred')
            @logger.error(e)
          end
        end
      end

      def start
        return unless @thread.nil?
        @running = true
        @thread = Thread.new do
          process
        end
      end

      def stop
        @running = false
      end

      def kill
        return if @thread.nil?
        @thread.kill
        @thread = nil
      end

      def wait
        @thread.join
      end
    end

    class PollerWorker < Worker
      def initialize(options = {})
        @states = options[:states]
        @activity_arn = options[:activity_arn]
        @sink = options[:sink]
        @activities = options[:activities]
        @max_retry = options[:max_retry]
        @logger = Logger.new(STDOUT)
      end

      def run
        activity_task = StepFunctions.with_retries(max_retry: @max_retry) do
          begin
            @states.get_activity_task(activity_arn: @activity_arn)
          rescue => e
```

```
            @logger.error('Failed to retrieve activity task')
            @logger.error(e)
          end
        end
        return if activity_task.nil? || activity_task.task_token.nil?
        @activities.add(activity_task.task_token)
        @sink.push(activity_task)
      end
    end

    class ActivityWorker < Worker
      def initialize(options = {})
        @states = options[:states]
        @block = options[:block]
        @sink = options[:sink]
        @activities = options[:activities]
        @max_retry = options[:max_retry]
        @logger = Logger.new(STDOUT)
      end

      def run
        activity_task = @sink.pop
        result = @block.call(JSON.parse(activity_task.input))
        send_task_success(activity_task, result)
      rescue => e
        send_task_failure(activity_task, e)
      ensure
        @activities.delete(activity_task.task_token) unless activity_task.nil?
      end

      def send_task_success(activity_task, result)
        StepFunctions.with_retries(max_retry: @max_retry) do
          begin
            @states.send_task_success(
              task_token: activity_task.task_token,
              output: JSON.dump(result)
            )
          rescue => e
            @logger.error('Failed to send task success')
            @logger.error(e)
          end
        end
      end
```

```
    def send_task_failure(activity_task, error)
      StepFunctions.with_retries do
        begin
          @states.send_task_failure(
            task_token: activity_task.task_token,
            cause: error.message
          )
        rescue => e
          @logger.error('Failed to send task failure')
          @logger.error(e)
        end
      end
    end
  end

  class HeartbeatWorker < Worker
    def initialize(options = {})
      @states = options[:states]
      @activities = options[:activities]
      @heartbeat_delay = options[:heartbeat_delay]
      @max_retry = options[:max_retry]
      @logger = Logger.new(STDOUT)
    end

    def run
      sleep(@heartbeat_delay)
      @activities.each do |token|
        send_heartbeat(token)
      end
    end

    def send_heartbeat(token)
      StepFunctions.with_retries(max_retry: @max_retry) do
        begin
          @states.send_task_heartbeat(token)
        rescue => e
          @logger.error('Failed to send heartbeat for activity')
          @logger.error(e)
        end
      end
    rescue => e
      @logger.error('Failed to send heartbeat for activity')
      @logger.error(e)
    end
```

```
    end
  end
end
```

Pilihan

Sebuah `Choice` state (`"Type": "Choice"`) menambahkan logika kondisional ke mesin negara.

Selain sebagian besar bidang negara umum, `Choice` negara bagian berisi bidang tambahan berikut.

`Choices` (Wajib)

Array Aturan Pilihan yang menentukan status mesin status yang akan ditransisikan ke berikutnya. Anda menggunakan operator perbandingan dalam Aturan Pilihan untuk membandingkan variabel input dengan nilai tertentu. Misalnya, menggunakan Aturan Pilihan Anda dapat membandingkan jika variabel input lebih besar dari atau kurang dari 100.

Ketika `Choice` negara dijalankan, itu mengevaluasi setiap Aturan Pilihan untuk benar atau salah. Berdasarkan hasil evaluasi ini, Langkah Fungsi transisi ke negara berikutnya dalam alur kerja.

Anda harus menentukan setidaknya satu aturan di `Choice` negara bagian.

`Default` (Opsional, Direkomendasikan)

Nama status yang akan ditransisikan jika tidak ada transisi di `Choices` yang diambil.

> **Important**
>
> Status `Choice` tidak mendukung bidang `End`. Selain itu, status tersebut hanya menggunakan `Next` di dalam bidang `Choices`nya.

> **Tip**
>
> Untuk menyebarkan contoh alur kerja yang menggunakan *`Choice`* status untuk AndaAkun AWS, lihat Modul 5 - Pilihan Negara dan Peta Negara Workshop. AWS Step Functions

Aturan Pilihan

Sebuah `Choice` negara harus memiliki `Choices` bidang yang nilainya adalah array non-kosong. Setiap elemen dalam array ini adalah objek yang disebut Choice Rule, yang berisi berikut ini:

- Perbandingan — Dua bidang yang menentukan variabel input untuk membandingkan tipe perbandingan, dan nilai untuk membandingkan variabel. Aturan Pilihan mendukung perbandingan antara dua variabel. Dalam Aturan Pilihan, nilai variabel dapat dibandingkan dengan nilai lain dari masukan negara dengan `Path` menambahkan nama operator perbandingan yang didukung. Nilai-nilai `Variable` dan bidang Path dalam perbandingan harus Jalur Referensi valid.
- Bidang **`Next`** — Nilai bidang ini harus cocok dengan nama status di mesin status.

Contoh berikut memeriksa apakah nilai numerik sama dengan 1.

```
{
  "Variable": "$.foo",
  "NumericEquals": 1,
  "Next": "FirstMatchState"
}
```

Contoh berikut memeriksa apakah string sama dengan `MyString`.

```
{
  "Variable": "$.foo",
  "StringEquals": "MyString",
  "Next": "FirstMatchState"
}
```

Contoh berikut memeriksa apakah string lebih besar dengan `MyStringABC`.

```
{
  "Variable": "$.foo",
  "StringGreaterThan": "MyStringABC",
  "Next": "FirstMatchState"
}
```

Contoh berikut memeriksa apakah string bernilai null.

```
{
```

```
 "Variable": "$.possiblyNullValue",
 "IsNull": true
}
```

Contoh berikut menunjukkan bagaimana StringEquals aturan hanya dievaluasi bila `$.keyThatMightNotExist` ada karena Aturan `IsPresent` Pilihan sebelumnya.

```
"And": [
 {
 "Variable": "$.keyThatMightNotExist",
 "IsPresent": true
 },
 {
 "Variable": "$.keyThatMightNotExist",
 "StringEquals": "foo"
 }
]
```

Contoh berikut memeriksa apakah pola dengan kecocokan wildcard.

```
{
 "Variable": "$.foo",
 "StringMatches": "log-*.txt"
}
```

Contoh berikut memeriksa apakah waktu stempel sama dengan `2001-01-01T12:00:00Z`.

```
{
  "Variable": "$.foo",
  "TimestampEquals": "2001-01-01T12:00:00Z",
  "Next": "FirstMatchState"
}
```

Contoh berikut membandingkan variabel dengan nilai lain dari input status.

```
{
 "Variable": "$.foo",
 "StringEqualsPath": "$.bar"
}
```

Step Functions meneliti masing-masing Aturan Pilihan dalam urutan yang terdaftar di bidang `Choices`. Lalu bertransisi ke status yang ditentukan di bidang `Next` Aturan Pilihan pertama saat variabel cocok dengan nilai yang sesuai dengan operator perbandingan.

Operator perbandingan berikut didukung:

- `And`
- `BooleanEquals`,`BooleanEqualsPath`
- `IsBoolean`
- `IsNull`
- `IsNumeric`
- `IsPresent`
- `IsString`
- `IsTimestamp`
- `Not`
- `NumericEquals`,`NumericEqualsPath`
- `NumericGreaterThan`,`NumericGreaterThanPath`
- `NumericGreaterThanEquals`,`NumericGreaterThanEqualsPath`
- `NumericLessThan`,`NumericLessThanPath`
- `NumericLessThanEquals`,`NumericLessThanEqualsPath`
- `Or`
- `StringEquals`,`StringEqualsPath`
- `StringGreaterThan`,`StringGreaterThanPath`
- `StringGreaterThanEquals`,`StringGreaterThanEqualsPath`
- `StringLessThan`,`StringLessThanPath`
- `StringLessThanEquals`,`StringLessThanEqualsPath`
- `StringMatches`
- `TimestampEquals`,`TimestampEqualsPath`
- `TimestampGreaterThan`,`TimestampGreaterThanPath`
- `TimestampGreaterThanEquals`,`TimestampGreaterThanEqualsPath`
- `TimestampLessThan`,`TimestampLessThanPath`
- `TimestampLessThanEquals`,`TimestampLessThanEqualsPath`

Untuk masing-masing operator ini, nilai yang sesuai harus dari tipe yang sesuai: string, nomor, Boolean, atau waktu stempel. Step Functions tidak mencoba untuk mencocokkan bidang numerik dengan nilai string. Namun, karena bidang waktu stempel secara logis adalah string, ada kemungkinan bahwa bidang yang dianggap sebagai stempel waktu dapat dicocokkan dengan pembanding `StringEquals`.

> **Note**
>
> Untuk interoperabilitas, jangan berasumsi bahwa perbandingan numerik bekerja dengan nilai di luar magnitudo atau presisi yang diwakili tipe data `binary64` 754-2008 IEEE. Secara khusus, bilangan bulat di luar jangkauan $[-2^{53}+1, 2^{53}-1]$ mungkin gagal untuk membandingkan dengan cara yang diharapkan.
> Stempel waktu (misalnya, `2016-08-18T17:33:00Z`) harus sesuai dengan Profil RFC3339 ISO 8601, dengan pembatasan lebih lanjut:
>
> - Huruf kapital `T` harus memisahkan bagian tanggal dan waktu.
> - Huruf kapital `Z` harus menunjukkan bahwa offset zona waktu numerik tidak ada.
>
> Untuk memahami perilaku perbandingan string, lihat Dokumentasi `compareTo` Java.
> Nilai operator `And` dan `Or` harus berupa array non-kosong dari Aturan Pilihan yang tidak boleh berisi bidang `Next`. Demikian juga, nilai operator `Not` harus menjadi Aturan Pilihan tunggal yang tidak boleh mengandung bidang `Next`.
> Anda dapat membuat Aturan Pilihan yang kompleks dan di-nest menggunakan `And`, `Not`, dan `Or`. Namun, bidang `Next` hanya dapat muncul di Aturan Pilihan tingkat atas.
> Perbandingan string terhadap pola dengan satu atau lebih wildcard (“*”) dapat dilakukan dengan operator StringMatches perbandingan. Karakter wildcard dihilangkan menggunakan `\\` (`Ex: “\\*”`) standar. Tidak ada karakter selain “*” yang memiliki arti khusus selama pencocokan.

Contoh Status Pilihan

Berikut ini adalah contoh status `Choice` dan status lain tempat status bertransisi.

> **Note**
>
> Anda harus menentukan bidang `$.type`. Jika input status tidak berisi bidang `$.type`, eksekusi gagal dan kesalahan ditampilkan di riwayat eksekusi. Anda hanya dapat

menentukan string di bidang `StringEquals` yang cocok dengan nilai literal. Misalnya, `"StringEquals": "Buy"`.

```
"ChoiceStateX": {
  "Type": "Choice",
  "Choices": [
    {
      "Not": {
        "Variable": "$.type",
        "StringEquals": "Private"
      },
      "Next": "Public"
    },
    {
      "Variable": "$.value",
      "NumericEquals": 0,
      "Next": "ValueIsZero"
    },
    {
      "And": [
        {
          "Variable": "$.value",
          "NumericGreaterThanEquals": 20
        },
        {
          "Variable": "$.value",
          "NumericLessThan": 30
        }
      ],
      "Next": "ValueInTwenties"
    }
  ],
  "Default": "DefaultState"
},

"Public": {
  "Type" : "Task",
  "Resource": "arn:aws:lambda:us-east-1:123456789012:function:Foo",
  "Next": "NextState"
},

"ValueIsZero": {
```

```
  "Type" : "Task",
  "Resource": "arn:aws:lambda:us-east-1:123456789012:function:Zero",
  "Next": "NextState"
},

"ValueInTwenties": {
  "Type" : "Task",
  "Resource": "arn:aws:lambda:us-east-1:123456789012:function:Bar",
  "Next": "NextState"
},

"DefaultState": {
  "Type": "Fail",
  "Cause": "No Matches!"
}
```

Dalam contoh ini, mesin status dimulai dengan nilai input berikut.

```
{
  "type": "Private",
  "value": 22
}
```

Step Functions bertransisi ke status `ValueInTwenties`, berdasarkan bidang `value`.

Jika tidak ada kecocokan untuk `Choices` status `Choice`, status yang disediakan di bidang `Default` berjalan sebagai gantinya. Jika status `Default` tidak ditentukan, eksekusi gagal dengan kesalahan.

Tunggu

Status `Wait` (`"Type": "Wait"`) menunda mesin status dari melanjutkan ke waktu yang ditentukan. Anda dapat memilih waktu relatif, ditentukan dalam detik sejak saat status dimulai, atau waktu akhir absolut, yang ditetapkan sebagai stempel waktu.

Selain bidang status umum, `Wait` memiliki satu bidang berikut.

`Seconds`

Waktu, dalam detik, untuk menunggu sebelum status yang ditentukan di `Next` dimulai. Anda harus menentukan waktu sebagai nilai integer positif dari 0 hingga 99999999.

`Timestamp`

Waktu absolut untuk menunggu sampai waktu awal status yang ditentukan di bidang `Next`.

Stempel waktu harus sesuai dengan profil RFC3339 ISO 8601, dengan pembatasan lebih lanjut bahwa huruf kapital T harus memisahkan bagian tanggal dan waktu, dan huruf kapital Z harus menunjukkan bahwa offset zona waktu numerik tidak ada, misalnya `2024-08-18T17:33:00Z`.

> **Note**
>
> Saat ini, jika Anda menentukan waktu tunggu sebagai stempel waktu, Step Functions mempertimbangkan nilai waktu hingga detik dan memotong milidetik.

`SecondsPath`

Waktu, dalam detik, untuk menunggu sebelum memulai status yang ditentukan di bidang `Next`, yang ditentukan menggunakan jalur dari data input status.

Anda harus menentukan nilai integer untuk bidang ini.

`TimestampPath`

Waktu absolut untuk menunggu sampai memulai status yang ditentukan di bidang `Next`, yang ditentukan menggunakan jalur dari data input status.

> **Note**
>
> Anda harus menentukan dengan tepat salah satu dari `Seconds`, `Timestamp`, `SecondsPath`, atau `TimestampPath`. Selain itu, waktu tunggu maksimum yang dapat Anda tentukan untuk Alur Kerja Standar dan alur kerja Ekspres masing-masing adalah satu tahun dan lima menit.

Contoh Status Tunggu

Status `Wait` berikut memperkenalkan penundaan 10 detik pada mesin status.

```
"wait_ten_seconds": {
```

```
  "Type": "Wait",
  "Seconds": 10,
  "Next": "NextState"
}
```

Dalam contoh berikutnya, `Wait` negara menunggu hingga waktu absolut: 14 Maret 2024, pukul 1:59 UTC.

```
"wait_until" : {
  "Type": "Wait",
  "Timestamp": "2024-03-14T01:59:00Z",
  "Next": "NextState"
}
```

Anda tidak perlu melakukan hard-coding pada durasi tunggu. Misalnya, diberikan data input berikut:

```
{
  "expirydate": "2024-03-14T01:59:00Z"
}
```

Anda dapat memilih nilai "expiydate" dari input menggunakan jalur referensi untuk memilihnya dari data input.

```
"wait_until" : {
    "Type": "Wait",
    "TimestampPath": "$.expirydate",
    "Next": "NextState"
}
```

Berhasil

Status `Succeed` (`"Type": "Succeed"`) berhasil menghentikan eksekusi. Status `Succeed` adalah target yang berguna untuk cabang status `Choice` yang tidak melakukan apa-apa kecuali menghentikan eksekusi.

Karena status `Succeed` adalah status terminal, status tersebut tidak memiliki `Next`, dan tidak memerlukan bidang `End`, seperti yang ditunjukkan dalam contoh berikut.

```
"SuccessState": {
```

```
  "Type": "Succeed"
}
```

Gagal

Status `Fail` (`"Type": "Fail"`) menghentikan eksekusi mesin status dan menandainya sebagai kegagalan, kecuali jika tertangkap oleh blok `Catch`.

Status `Fail` hanya mengizinkan penggunaan bidang `Type` dan `Comment` dari set bidang status umum. Selain itu, status `Fail` mengizinkan bidang berikut.

`Cause` (Opsional)

Sebuah string khusus yang menjelaskan penyebab kesalahan. Anda dapat menentukan bidang ini untuk tujuan operasional atau diagnostik.

`CausePath` (Opsional)

Jika Anda ingin memberikan deskripsi rinci tentang penyebab kesalahan secara dinamis dari input status menggunakan.jalur referensi, gunakan`CausePath`. Ketika diselesaikan, jalur referensi harus memilih bidang yang berisi nilai string.

Anda juga dapat menentukan`CausePath`menggunakan sebuahfungsi intrinsikyang mengembalikan string. Intrinsik ini adalah:Format Negara,States.JsonToString,States.ArrayGetItem,States.Base64Encode,States.Base64Decode,Negara bagian.Hash, dan States.UUID.

> **Important**
>
> - Anda dapat menentukan salah satu`Cause`atau`CausePath`, tetapi tidak keduanya dalam definisi status Gagal Anda.
> - Sebagai praktik terbaik keamanan informasi, kami sarankan Anda menghapus informasi sensitif atau detail sistem internal dari deskripsi penyebab.

`Error` (Opsional)

Nama kesalahan yang dapat Anda berikan untuk melakukan penanganan kesalahan menggunakanCoba lagiatauTangkapladang. Anda juga dapat memberikan nama kesalahan untuk tujuan operasional atau diagnostik.

`ErrorPath` (Opsional)

Jika Anda ingin memberikan nama untuk kesalahan secara dinamis dari input status menggunakanjalur referensi, gunakan`ErrorPath`. Ketika diselesaikan, jalur referensi harus memilih bidang yang berisi nilai string.

Anda juga dapat menentukan`ErrorPath`menggunakan sebuahfungsi intrinsikyang mengembalikan string. Intrinsik ini adalah:Format Negara,States.JsonToString,States.ArrayGetItem,States.Base64Encode,States.Base64Decode,Negara bagian.Hash, danStates.UUID.

> **Important**
>
> - Anda dapat menentukan salah satu`Error`atau`ErrorPath`, tetapi tidak keduanya dalam definisi status Gagal Anda.
> - Sebagai praktik terbaik keamanan informasi, kami sarankan Anda menghapus informasi sensitif atau detail sistem internal dari nama kesalahan.

Karena status `Fail` selalu keluar dari mesin status, status tersebut tidak memiliki bidang `Next` dan tidak memerlukan bidang `End`.

Contoh definisi status gagal

Contoh definisi status Gagal berikut menentukan statis`Error`dan`Cause`nilai bidang.

```
"FailState": {
  "Type": "Fail",
  "Cause": "Invalid response.",
  "Error": "ErrorA"
}
```

Contoh definisi status Fail berikut menggunakan jalur referensi secara dinamis untuk menyelesaikan`Error`dan`Cause`nilai bidang.

```
"FailState": {
  "Type": "Fail",
  "CausePath": "$.Cause",
  "ErrorPath": "$.Error"
```

```
}
```

Contoh definisi status Gagal berikut menggunakanFormat Negarafungsi intrinsik untuk menentukan`Error`dan`Cause`nilai bidang secara dinamis.

```
"FailState": {
  "Type": "Fail",
  "CausePath": "States.Format('This is a custom error message for {}, caused by {}.',
 $.Error, $.Cause)",
  "ErrorPath": "States.Format('{}', $.Error)"
}
```

Paralel

`Parallel`State (`"Type": "Parallel"`) dapat digunakan untuk menambahkan cabang eksekusi terpisah di mesin negara Anda.

Selain bidang status umum, status `Parallel` memperkenalkan bidang tambahan ini.

`Branches` (Wajib)

Array objek yang menentukan mesin status untuk mengeksekusi secara paralel. Setiap objek mesin status tersebut harus memiliki bidang dengan nama `States` dan `StartAt`, yang maknanya persis seperti yang ada di tingkat atas mesin status.

`ResultPath` (Opsional)

Menentukan tempat (dalam input) untuk meletakkan output cabang. Input kemudian difilter sebagaimana ditentukan oleh bidang `OutputPath` (jika ada) sebelum digunakan sebagai output status. Untuk informasi selengkapnya, lihat Pemrosesan Input dan Output.

`ResultSelector` (Opsional)

Teruskan koleksi pasangan nilai kunci, tempat nilai-nilai bersifat statis atau dipilih dari hasil. Untuk informasi selengkapnya, lihat ResultSelector.

`Retry` (Opsional)

Array objek, yang disebut Retriers, yang menentukan kebijakan percobaan ulang jika status menemukan kesalahan waktu aktif. Untuk informasi selengkapnya, lihat Nyatakan contoh mesin menggunakan Coba Ulang dan menggunakan Catch.

`Catch` (Opsional)

Array objek, yang disebut Catcher, yang menentukan status fallback yang dijalankan jika status menemukan kesalahan waktu aktif dan kebijakan percobaan ulang habis atau tidak ditentukan. Untuk informasi selengkapnya, lihat Status Fallback.

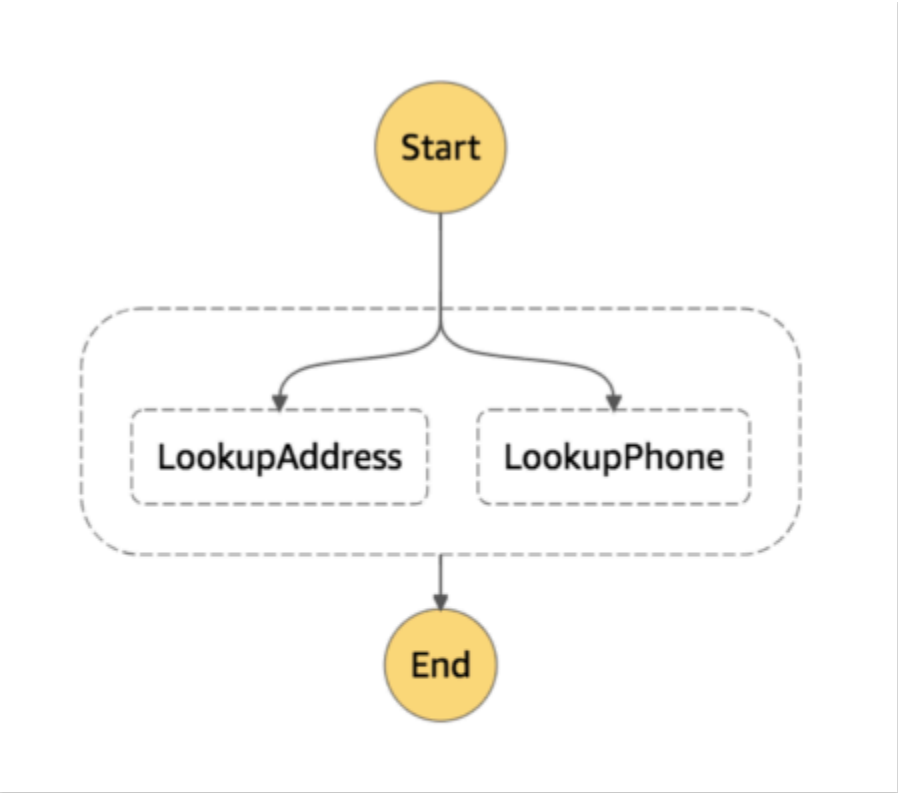
Sebuah `Parallel` status menyebabkan AWS Step Functions untuk mengeksekusi setiap cabang, dimulai dengan status yang disebutkan di `StartAt` bidang cabang itu, serentetan mungkin, dan menunggu sampai semua cabang berakhir (mencapai status terminal) sebelum memproses bidang `Parallel` status. `Next`

Contoh Status Paralel

```
{
  "Comment": "Parallel Example.",
  "StartAt": "LookupCustomerInfo",
  "States": {
    "LookupCustomerInfo": {
      "Type": "Parallel",
      "End": true,
      "Branches": [
        {
         "StartAt": "LookupAddress",
         "States": {
           "LookupAddress": {
             "Type": "Task",
             "Resource":
               "arn:aws:lambda:us-east-1:123456789012:function:AddressFinder",
             "End": true
           }
         }
       },
       {
         "StartAt": "LookupPhone",
         "States": {
           "LookupPhone": {
             "Type": "Task",
             "Resource":
               "arn:aws:lambda:us-east-1:123456789012:function:PhoneFinder",
             "End": true
           }
         }
```

```
        }
       ]
     }
   }
}
```

Dalam contoh ini, cabang `LookupAddress` dan `LookupPhone` dieksekusi secara paralel. Berikut adalah bagaimana alur kerja visual terlihat di konsol Step Functions.



Setiap cabang harus mandiri. Status di cabang status `Parallel` tidak boleh memiliki bidang `Next` yang menargetkan sebuah bidang di luar cabang tersebut, begitu pula status lain di luar cabang tidak boleh bertransisi ke cabang tersebut.

Pemrosesan Input dan Output Status Paralel

Status `Parallel` menyediakan setiap cabang dengan salinan data inputnya sendiri (tunduk pada modifikasi oleh bidang `InputPath`). Ini menghasilkan output yang merupakan array dengan satu elemen untuk setiap cabang, yang berisi output dari cabang tersebut. Tidak ada persyaratan bahwa semua elemen harus dengan tipe yang sama. Array output dapat dimasukkan ke dalam data input (dan seluruhnya dikirim sebagai output status `Parallel`) dengan menggunakan bidang `ResultPath` dengan cara umum (lihat Pemrosesan Input dan Output).

```
{
  "Comment": "Parallel Example.",
  "StartAt": "FunWithMath",
  "States": {
    "FunWithMath": {
      "Type": "Parallel",
      "End": true,
      "Branches": [
        {
          "StartAt": "Add",
          "States": {
            "Add": {
              "Type": "Task",
              "Resource": "arn:aws:states:us-east-1:123456789012:activity:Add",
              "End": true
            }
          }
        },
        {
          "StartAt": "Subtract",
          "States": {
            "Subtract": {
              "Type": "Task",
              "Resource": "arn:aws:states:us-east-1:123456789012:activity:Subtract",
              "End": true
            }
          }
        }
      ]
    }
  }
}
```

Jika status `FunWithMath` diberikan array `[3, 2]` sebagai input, maka kedua status `Add` dan `Subtract` menerima array tersebut sebagai masukan. Output dari `Subtract` tugas `Add` dan akan menjadi jumlah dan perbedaan antara elemen array 3 dan 2, yaitu `5` dan`1`, sedangkan output dari `Parallel` keadaan akan menjadi array.

```
[ 5, 1 ]
```

> **Tip**
>
> Jika status Paralel atau Peta yang Anda gunakan di mesin status Anda mengembalikan array array, Anda dapat mengubahnya menjadi array datar dengan ResultSelector bidang. Untuk informasi selengkapnya, lihat Meratakan array array.

Penanganan Kesalahan

Jika setiap cabang gagal karena kesalahan tidak tertangani atau dengan transisi ke status `Fail`, seluruh status `Parallel` dianggap telah gagal dan semua cabangnya dihentikan. Jika kesalahan tidak ditangani oleh status `Parallel`itu sendiri, Step Functions menghentikan eksekusi dengan kesalahan.

> **Note**
>
> Ketika status paralel gagal, fungsi Lambda Invoke terus berjalan dan pekerja aktivitas yang memproses token tugas tidak dihentikan.
>
> - Untuk menghentikan aktivitas yang berjalan lama, gunakan heartbeat untuk mendeteksi apakah cabangnya telah dihentikan oleh Step Functions, dan menghentikan pekerja yang memproses tugas. Memanggil `SendTaskHeartbeat`, `SendTaskSuccess`, atau `SendTaskFailure` akan mengembalikan kesalahan jika status telah gagal. Lihat Kesalahan Heartbeat.
> - Menjalankan fungsi Lambda tidak dapat dihentikan. Jika Anda telah menerapkan fallback, gunakan status `Wait` sehingga pekerjaan pembersihan terjadi setelah fungsi Lambda telah selesai.

Map

Gunakan `Map` status untuk menjalankan serangkaian langkah alur kerja untuk setiap item dalam kumpulan data. Iterasi `Map` status berjalan secara paralel, yang memungkinkan untuk memproses kumpulan data dengan cepat. `Map`status dapat menggunakan berbagai jenis input, termasuk array JSON, daftar objek Amazon S3, atau file CSV.

Step Functions menyediakan dua jenis mode pemrosesan untuk menggunakan `Map` status dalam alur kerja Anda: Mode sebaris dan mode Terdistribusi.

Untuk informasi tentang mode ini, dan cara menggunakan `Map` status di salah satu mode, lihat topik berikut:

- Mode pemrosesan status peta
  - Menggunakan status Peta dalam mode Inline
  - Menggunakan status Peta dalam mode Terdistribusi

> **Tip**
>
> Untuk menerapkan contoh alur kerja yang menggunakan *`Map`* status ke AndaAkun AWS, lihat Modul 5 - Status Pilihan dan Status Peta Lokakarya. AWS Step Functions

Mode pemrosesan status peta

Step Functions menyediakan mode pemrosesan berikut untuk `Map` status tergantung pada bagaimana Anda ingin memproses item dalam kumpulan data.

- Inline - Mode konkurensi terbatas. Dalam mode ini, setiap iterasi `Map` status berjalan dalam konteks alur kerja yang berisi status. `Map` Step Functions menambahkan riwayat eksekusi iterasi ini ke riwayat eksekusi alur kerja induk. Secara default, `Map` status berjalan dalam mode Inline.

  Dalam mode ini, `Map` status hanya menerima array JSON sebagai input. Selain itu, mode ini mendukung hingga 40 iterasi bersamaan.

  Untuk informasi selengkapnya, lihat Menggunakan status Peta dalam mode Inline.
- Didistribusikan - Mode konkurensi tinggi. Dalam mode ini, `Map` status menjalankan setiap iterasi sebagai eksekusi alur kerja anak, yang memungkinkan konkurensi tinggi hingga 10.000 eksekusi

alur kerja anak paralel. Setiap eksekusi alur kerja anak memiliki riwayat eksekusi terpisah sendiri dari alur kerja induk.

Dalam mode ini, `Map` status dapat menerima array JSON atau sumber data Amazon S3, seperti file CSV, sebagai inputnya.

Untuk informasi selengkapnya, lihat Menggunakan status Peta dalam mode Terdistribusi.

Mode yang harus Anda gunakan tergantung pada bagaimana Anda ingin memproses item dalam kumpulan data. Gunakan `Map` status dalam mode Inline jika riwayat eksekusi alur kerja Anda tidak akan melebihi 25.000 entri, atau jika Anda tidak memerlukan lebih dari 40 iterasi bersamaan.

Gunakan `Map` status dalam mode Terdistribusi saat Anda perlu mengatur beban kerja paralel skala besar yang memenuhi kombinasi kondisi berikut:

- Ukuran dataset Anda melebihi 256 KB.
- Riwayat peristiwa eksekusi alur kerja melebihi 25.000 entri.
- Anda memerlukan konkurensi lebih dari 40 iterasi paralel.

Topik

- Mode sebaris dan perbedaan mode Terdistribusi
- Menggunakan status Peta dalam mode Inline
- Menggunakan status Peta dalam mode Terdistribusi untuk mengatur beban kerja paralel skala besar

Mode sebaris dan perbedaan mode Terdistribusi

Tabel berikut menyoroti perbedaan antara mode Inline dan Distributed.

Modus sebaris	Mode terdistribusi
Supported data sources	
Menerima array JSON diteruskan dari langkah sebelumnya dalam alur kerja sebagai input.	Menerima sumber data berikut sebagai masukan:

Modus sebaris	Mode terdistribusi
	• Array JSON diteruskan dari langkah sebelumnya dalam alur kerja • JSON dalam bucket Amazon S3 yang berisi array • File CSV dalam ember Amazon S3 • Daftar objek Amazon S3 • Inventaris Amazon S3
Map iterations	
Dalam mode ini, setiap iterasi `Map` status berjalan dalam konteks alur kerja yang berisi status. `Map` Step Functions menambahkan riwayat eksekusi iterasi ini ke riwayat eksekusi alur kerja induk.	Dalam mode ini, `Map` status menjalankan setiap iterasi sebagai eksekusi alur kerja anak, yang memungkinkan konkurensi tinggi hingga 10.000 eksekusi alur kerja anak paralel. Setiap eksekusi alur kerja anak memiliki riwayat eksekusi terpisah sendiri dari alur kerja induk.
Maximum concurrency for parallel iterations	
Memungkinkan Anda menjalankan hingga 40 iterasi secara bersamaan mungkin.	Memungkinkan Anda menjalankan hingga 10.000 eksekusi alur kerja anak paralel untuk memproses jutaan item data sekaligus.
Input payload and event history sizes	
Memberlakukan batas 256 KB pada ukuran muatan input dan 25.000 entri dalam riwayat peristiwa eksekusi.	Memungkinkan Anda mengatasi batasan ukuran payload karena `Map` status dapat membaca input langsung dari sumber data Amazon S3. Dalam mode ini, Anda juga dapat mengatasi batasan riwayat eksekusi karena eksekusi alur kerja anak yang dimulai oleh `Map` status mempertahankan histori eksekusi mereka sendiri dan terpisah dari riwayat eksekusi alur kerja induk.

Modus sebaris	Mode terdistribusi
Monitoring and observability	
Anda dapat meninjau riwayat eksekusi alur kerja dari konsol atau dengan menjalankan tindakan `GetExecutionHistory` API. Anda juga dapat melihat riwayat eksekusi melalui CloudWatch dan X-Ray.	Saat Anda menjalankan `Map` status dalam mode Distributed, Step Functions akan membuat resource Map Run. Map Run mengacu pada sekumpulan eksekusi alur kerja anak yang memulai status Peta Terdistri busi. Anda dapat melihat Map Run di konsol Step Functions. Anda juga dapat menjalankan tindakan `DescribeMapRun` API. Map Run juga memancarkan metrik ke. CloudWatch Untuk informasi selengkapnya, lihat Memeriksa Peta Jalankan eksekusi status Peta Terdistri busi.

Menggunakan status Peta dalam mode Inline

Secara default, `Map` status berjalan dalam mode Inline. Dalam mode Inline, status Peta hanya menerima array JSON sebagai input. Ia menerima array ini dari langkah sebelumnya dalam alur kerja. Dalam mode ini, setiap iterasi `Map` status berjalan dalam konteks alur kerja yang berisi status. `Map` Step Functions menambahkan riwayat eksekusi iterasi ini ke riwayat eksekusi alur kerja induk.

Dalam mode ini, `Map` status mendukung hingga 40 iterasi bersamaan.

`Map`Status yang disetel ke Inline dikenal sebagai status Peta Inline. Gunakan `Map` status dalam mode Inline jika riwayat eksekusi alur kerja Anda tidak akan melebihi 25.000 entri, atau jika Anda tidak memerlukan lebih dari 40 iterasi bersamaan.

Untuk pengenalan menggunakan status Peta Inline, lihat tutorialnyaUlangi tindakan menggunakan status Peta Inline.

Daftar Isi

- Konsep kunci dalam topik ini
- Bidang status Peta Inline
- Bidang usang

- Contoh status Peta Sebaris
- Contoh status Peta Sebaris dengan ItemSelector
- Pemrosesan input dan output Map status sebaris

Konsep kunci dalam topik ini

Modus sebaris

Mode konkurensi terbatas dari negara. `Map` Dalam mode ini, setiap iterasi `Map` status berjalan dalam konteks alur kerja yang berisi status. `Map` Step Functions menambahkan riwayat eksekusi iterasi ini ke riwayat eksekusi alur kerja induk. `Map`status berjalan dalam mode Inline secara default.

Mode ini hanya menerima array JSON sebagai input dan mendukung hingga 40 iterasi bersamaan.

Status Peta Sebaris

`Map`Status diatur ke mode Inline.

Alur kerja peta

Kumpulan langkah-langkah yang dijalankan oleh `Map` state untuk setiap iterasi.

Iterasi status peta

Pengulangan alur kerja yang didefinisikan di dalam negara. `Map`

Bidang status Peta Inline

Untuk menggunakan status Peta Sebaris dalam alur kerja Anda, tentukan satu atau beberapa bidang ini. Anda menentukan bidang ini selain bidang status umum.

`Type` (Wajib)

Menetapkan jenis negara, sepertiMap.

`ItemProcessor` (Wajib)

Berisi objek JSON berikut yang menentukan mode pemrosesan `Map` status dan definisi.

Definisi tersebut berisi serangkaian langkah yang harus diulang untuk memproses setiap item array.

- `ProcessorConfig`— Objek JSON opsional yang menentukan mode pemrosesan untuk negara. `Map` Objek ini berisi `Mode` sub-bidang. Bidang ini default ke`INLINE`, yang menggunakan `Map` status dalam mode Inline.

  Dalam mode ini, kegagalan iterasi apa pun menyebabkan `Map` status gagal. Semua iterasi berhenti ketika `Map` status gagal.
- `StartAt`- Menentukan string yang menunjukkan keadaan pertama dalam alur kerja. String ini peka huruf besar/kecil dan harus cocok dengan nama salah satu objek state. Status ini berjalan pertama kali untuk setiap item dalam kumpulan data. Masukan eksekusi apa pun yang Anda berikan ke `Map` status akan diteruskan ke `StartAt` status terlebih dahulu.
- `States`— Objek JSON yang berisi kumpulan status yang dibatasi koma. Dalam objek ini, Anda mendefinisikanMap workflow.

  > **Note**
  >
  > - Negara-negara di `ItemProcessor` lapangan hanya dapat bertransisi satu sama lain. Tidak ada negara di luar `ItemProcessor` lapangan yang dapat beralih ke keadaan di dalamnya.
  > - `ItemProcessor`Bidang menggantikan bidang yang sekarang tidak digunakan lagi. `Iterator` Meskipun Anda dapat terus menyertakan `Map` status yang menggunakan `Iterator` bidang ini, kami sangat menyarankan Anda mengganti bidang ini dengan`ItemProcessor`.
  >
  >   Step Functions Local saat ini tidak mendukung `ItemProcessor` bidang. Kami menyarankan Anda menggunakan `Iterator` bidang dengan Step Functions Local.

`ItemsPath` (Opsional)

Menentukan jalur referensi menggunakan JsonPathsintaks. Path ini memilih node JSON yang berisi array item di dalam input status. Untuk informasi selengkapnya, lihat ItemsPath.

`ItemSelector` (Opsional)

Mengganti nilai item array input sebelum diteruskan ke setiap iterasi `Map` status.

Di bidang ini, Anda menentukan JSON valid yang berisi kumpulan pasangan kunci-nilai. Pasangan ini dapat berisi salah satu dari berikut ini:

- Nilai statis yang Anda tentukan dalam definisi mesin status Anda.
- Nilai yang dipilih dari input status menggunakan jalur.

- Nilai diakses dari objek konteks.

Untuk informasi selengkapnya, lihat ItemSelector.

`ItemSelector`Bidang menggantikan bidang yang sekarang tidak digunakan lagi. `Parameters` Meskipun Anda dapat terus menyertakan `Map` status yang menggunakan `Parameters` bidang ini, kami sangat menyarankan Anda mengganti bidang ini dengan`ItemSelector`.

`MaxConcurrency` (Opsional)

Menentukan nilai integer yang menyediakan batas atas pada jumlah iterasi `Map` negara yang dapat berjalan secara paralel. Misalnya, `MaxConcurrency` nilai 10 membatasi `Map` status menjadi 10 iterasi bersamaan yang berjalan pada satu waktu.

> **Note**
>
> Perulangan bersamaan mungkin terbatas. Ketika ini terjadi, beberapa iterasi tidak akan dimulai sampai iterasi sebelumnya selesai. Kemungkinan hal ini terjadi meningkat ketika array input Anda memiliki lebih dari 40 item.
> Untuk mencapai konkurensi yang lebih tinggi, pertimbangkanMenggunakan status Peta dalam mode Terdistribusi.

Nilai defaultnya adalah`0`, yang tidak membatasi konkurensi. Step Functions memanggil iterasi serempak mungkin.

`MaxConcurrency`Nilai dari `1` memanggil `ItemProcessor` sekali untuk setiap elemen array. Item dalam array diproses dalam urutan penampilan mereka dalam input. Step Functions tidak memulai iterasi baru sampai menyelesaikan iterasi sebelumnya.

`MaxConcurrencyPath` (Opsional)

Jika Anda ingin memberikan nilai konkurensi maksimum secara dinamis dari input status menggunakan jalur referensi, gunakan. `MaxConcurrencyPath` Ketika diselesaikan, jalur referensi harus memilih bidang yang nilainya adalah bilangan bulat non-negatif.

> **Note**
>
> Suatu `Map` negara tidak dapat mencakup keduanya `MaxConcurrency` dan`MaxConcurrencyPath`.

`ResultPath` (Opsional)

Menentukan di mana di input untuk menyimpan output dari iterasi `Map` negara. Status Peta kemudian menyaring input seperti yang ditentukan oleh `OutputPath`bidang, jika ditentukan. Kemudian, ia menggunakan input yang difilter sebagai output negara. Untuk informasi selengkapnya, lihat Pemrosesan Input dan Output.

`ResultSelector` (Opsional)

Lewati kumpulan pasangan nilai kunci, di mana nilainya statis atau dipilih dari hasilnya. Untuk informasi selengkapnya, lihat ResultSelector.

> **Tip**
>
> Jika status Paralel atau Peta yang Anda gunakan di mesin status Anda mengembalikan array array, Anda dapat mengubahnya menjadi array datar dengan ResultSelector bidang. Untuk informasi selengkapnya, lihat Meratakan array array.

`Retry` (Opsional)

Array objek, yang disebut Retriers, yang mendefinisikan kebijakan coba lagi. Status menggunakan kebijakan coba lagi ketika mereka menemukan kesalahan runtime. Untuk informasi selengkapnya, lihat Nyatakan contoh mesin menggunakan Coba Ulang dan menggunakan Catch.

> **Note**
>
> Jika Anda mendefinisikan Retrier untuk status Peta Sebaris, kebijakan coba lagi berlaku untuk semua iterasi `Map` status, bukan hanya iterasi yang gagal. Misalnya, `Map` status Anda berisi dua iterasi yang berhasil dan satu iterasi gagal. Jika Anda telah menentukan `Retry` bidang untuk `Map` status, kebijakan coba lagi berlaku untuk ketiga iterasi `Map` status, bukan hanya iterasi yang gagal.

`Catch` (Opsional)

Array objek, disebut Catch yang menentukan status fallback. Status menjalankan penangkap jika mereka mengalami kesalahan runtime dan tidak memiliki kebijakan coba lagi, atau kebijakan coba ulang mereka habis. Untuk informasi selengkapnya, lihat Status Fallback.

Bidang usang

> **Note**
>
> Meskipun Anda dapat terus menyertakan `Map` status yang menggunakan bidang berikut, kami sangat menyarankan Anda mengganti `Iterator` dengan `ItemProcessor` dan `Parameters` dengan`ItemSelector`.

`Iterator`

Menentukan objek JSON yang mendefinisikan satu set langkah-langkah yang memproses setiap elemen array.

`Parameters`

Menentukan koleksi pasangan kunci-nilai, di mana nilai-nilai dapat berisi salah satu dari berikut:

- Nilai statis yang Anda tentukan dalam definisi mesin status Anda.
- Nilai yang dipilih dari input menggunakan jalur.

Contoh status Peta Sebaris

Pertimbangkan data input berikut untuk `Map` status yang berjalan dalam mode Inline.

```
{
  "ship-date": "2016-03-14T01:59:00Z",
  "detail": {
    "delivery-partner": "UQS",
    "shipped": [
      { "prod": "R31", "dest-code": 9511, "quantity": 1344 },
      { "prod": "S39", "dest-code": 9511, "quantity": 40 },
      { "prod": "R31", "dest-code": 9833, "quantity": 12 },
      { "prod": "R40", "dest-code": 9860, "quantity": 887 },
      { "prod": "R40", "dest-code": 9511, "quantity": 1220 }
    ]
  }
}
```

Mengingat masukan sebelumnya, `Map` status dalam contoh berikut memanggil AWS Lambda fungsi bernama `ship-val` sekali untuk setiap item array di `shipped` bidang.

```
"Validate All": {
    "Type": "Map",
    "InputPath": "$.detail",
    "ItemProcessor": {
        "ProcessorConfig": {
            "Mode": "INLINE"
        },
        "StartAt": "Validate",
        "States": {
            "Validate": {
                "Type": "Task",
                "Resource": "arn:aws:states:::lambda:invoke",
                "OutputPath": "$.Payload",
                "Parameters": {
                    "FunctionName": "arn:aws:lambda:us-
east-2:123456789012:function:ship-val:$LATEST"
                },
                "End": true
            }
        }
    },
    "End": true,
    "ResultPath": "$.detail.shipped",
    "ItemsPath": "$.shipped"
}
```

Setiap iterasi `Map` status mengirimkan item dalam array, dipilih dengan `ItemsPath`bidang, sebagai input ke fungsi `ship-val` Lambda. Nilai-nilai berikut adalah contoh masukan yang dikirimkan `Map` status ke pemanggilan fungsi Lambda:

```
{
  "prod": "R31",
  "dest-code": 9511,
  "quantity": 1344
}
```

Setelah selesai, output status `Map` adalah array JSON, saat setiap item adalah output dari perulangan. Dalam hal ini, array ini berisi output dari fungsi `ship-val` Lambda.

Contoh status Peta Sebaris dengan **`ItemSelector`**

Misalkan fungsi `ship-val` Lambda pada contoh sebelumnya juga membutuhkan informasi tentang kurir pengiriman. Informasi ini merupakan tambahan untuk item dalam array untuk setiap iterasi. Anda dapat menyertakan informasi dari input, bersama dengan informasi khusus untuk iterasi `Map` status saat ini. Perhatikan `ItemSelector` bidang dalam contoh berikut:

```
"Validate-All": {
  "Type": "Map",
  "InputPath": "$.detail",
  "ItemsPath": "$.shipped",
  "MaxConcurrency": 0,
  "ResultPath": "$.detail.shipped",
  "ItemSelector": {
    "parcel.$": "$$.Map.Item.Value",
    "courier.$": "$.delivery-partner"
  },
  "ItemProcessor": {
    "StartAt": "Validate",
    "States": {
      "Validate": {
        "Type": "Task",
 "Resource": "arn:aws:lambda:us-east-1:123456789012:function:ship-val",
        "End": true
      }
    }
  },
  "End": true
}
```

`ItemSelector`Blok menggantikan input ke iterasi dengan node JSON. Node ini berisi data item saat ini dari objek konteks dan informasi kurir dari `delivery-partner` bidang input `Map` status. Berikut ini adalah contoh input ke iterasi tunggal. `Map`Status meneruskan input ini ke pemanggilan fungsi Lambda`ship-val`.

```
{
  "parcel": {
    "prod": "R31",
    "dest-code": 9511,
    "quantity": 1344
   },
   "courier": "UQS"
```

```
}
```

Dalam contoh status Peta Inline sebelumnya, `ResultPath` bidang menghasilkan output dalam format yang sama dengan input. Namun, ia menimpa `detail.shipped` bidang dengan array di mana setiap elemen adalah output dari setiap pemanggilan Lambda `ship-val` iterasi.

Untuk informasi selengkapnya tentang penggunaan status Peta Sebaris dan bidangnya, lihat berikut ini.

- Ulangi tindakan menggunakan status Peta Inline
- Pengolahan Input dan Output di Step Functions
- ItemsPath
- Data Objek Konteks untuk Status Peta

Pemrosesan input dan output **Map** status sebaris

Untuk `Map` status tertentu, `InputPath`pilih subset dari masukan negara.

Masukan dari sebuah `Map` negara harus menyertakan array JSON. `Map`Status menjalankan `ItemProcessor` bagian satu kali untuk setiap item dalam array. Jika Anda menentukan `ItemsPath`bidang, `Map` status memilih di mana di input untuk menemukan array untuk diulang. Jika tidak ditentukan, nilai `ItemsPath` berupa `$`, dan bagian `ItemProcessor` berharap agar array adalah satu-satunya input. Jika Anda menentukan `ItemsPath` bidang, nilainya harus berupa Jalur Referensi. `Map`Status menerapkan jalur ini ke input efektif setelah menerapkan`InputPath`. `ItemsPath`Harus mengidentifikasi bidang yang nilainya adalah array JSON.

Masukan untuk setiap iterasi, secara default, adalah elemen tunggal dari bidang array yang diidentifikasi oleh `ItemsPath` nilai. Anda dapat mengganti nilai ini dengan `ItemSelector` bidang.

Setelah selesai, output status `Map` adalah array JSON, saat setiap item adalah output dari perulangan.

Untuk informasi selengkapnya tentang input dan output status Peta Inline, lihat berikut ini:

- Ulangi tindakan menggunakan status Peta Inline
- Contoh status Peta Sebaris dengan `ItemSelector`
- Pengolahan Input dan Output di Step Functions

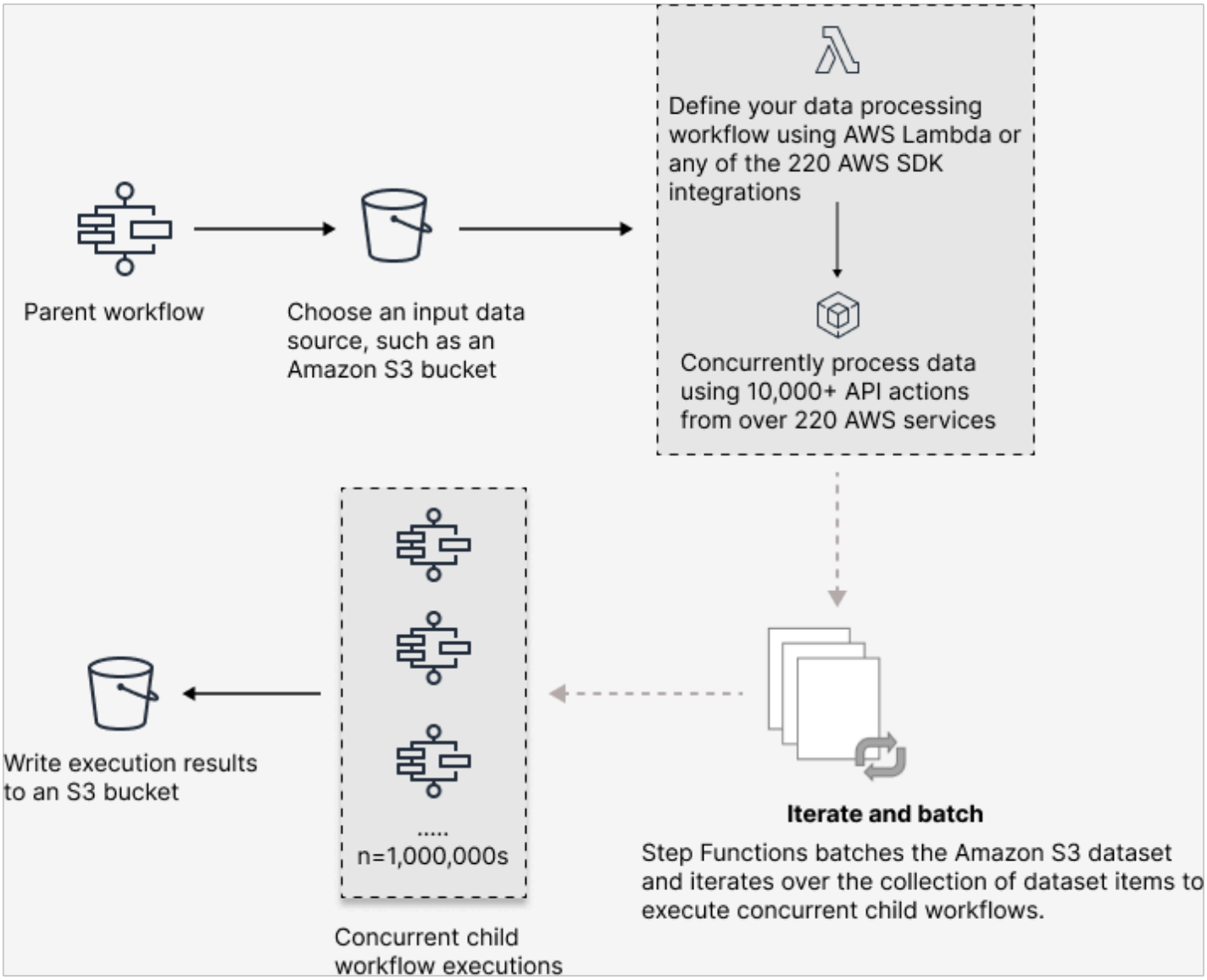
- Data Objek Konteks untuk Status Peta
- Secara dinamis memproses data dengan status Peta

Menggunakan status Peta dalam mode Terdistribusi untuk mengatur beban kerja paralel skala besar

Dengan Step Functions, Anda dapat mengatur beban kerja paralel skala besar untuk melakukan tugas, seperti pemrosesan data semi-terstruktur berdasarkan permintaan. Beban kerja paralel ini memungkinkan Anda memproses sumber data skala besar yang disimpan di Amazon S3 secara bersamaan. Misalnya, Anda dapat memproses satu file JSON atau CSV yang berisi sejumlah besar data. Atau Anda dapat memproses satu set besar objek Amazon S3.

Untuk menyiapkan beban kerja paralel skala besar dalam alur kerja Anda, sertakan `Map` status dalam mode Terdistribusi. Status Peta memproses item dalam kumpulan data secara bersamaan. `Map`Status yang disetel ke Distributed dikenal sebagai status Peta Terdistribusi. Dalam mode Terdistribusi, `Map` status memungkinkan pemrosesan konkurensi tinggi. Dalam mode Terdistribusi, `Map` status memproses item dalam kumpulan data dalam iterasi yang disebut eksekusi alur kerja anak. Anda dapat menentukan jumlah eksekusi alur kerja anak yang dapat berjalan secara paralel. Setiap eksekusi alur kerja anak memiliki riwayat eksekusi terpisah sendiri dari alur kerja induk. Jika Anda tidak menentukan, Step Functions menjalankan 10.000 eksekusi alur kerja anak paralel secara paralel.

Ilustrasi berikut menjelaskan bagaimana Anda dapat mengatur beban kerja paralel skala besar dalam alur kerja Anda.



Dalam topik ini:

- Istilah kunci
- Contoh definisi status Peta Terdistribusi
- Izin untuk menjalankan Peta Terdistribusi
- Bidang status Peta Terdistribusi
- Langkah selanjutnya

Istilah kunci

Mode terdistribusi

Mode pemrosesan status Peta. Dalam mode ini, setiap iterasi `Map` status berjalan sebagai eksekusi alur kerja anak yang memungkinkan konkurensi tinggi. Setiap eksekusi alur kerja anak memiliki riwayat eksekusi sendiri, yang terpisah dari riwayat eksekusi alur kerja induk. Mode ini mendukung input pembacaan dari sumber data Amazon S3 skala besar.

Status Peta Terdistribusi

Status Peta diatur ke mode pemrosesan Terdistribusi.

Alur kerja peta

Serangkaian langkah yang dijalankan oleh suatu `Map` negara.

Alur kerja orang tua

Alur kerja yang berisi satu atau lebih status Peta Terdistribusi.

Eksekusi alur kerja anak

Iterasi dari status Peta Terdistribusi. Eksekusi alur kerja anak memiliki riwayat eksekusi sendiri, yang terpisah dari riwayat eksekusi alur kerja induk.

Peta Jalankan

Saat Anda menjalankan `Map` status dalam mode Distributed, Step Functions akan membuat resource Map Run. Map Run mengacu pada sekumpulan eksekusi alur kerja anak yang memulai status Peta Terdistribusi, dan pengaturan runtime yang mengontrol eksekusi ini. Step Functions menetapkan Amazon Resource Name (ARN) ke Map Run Anda. Anda dapat memeriksa Map Run di konsol Step Functions. Anda juga dapat menjalankan tindakan `DescribeMapRun` API. Map Run juga memancarkan metrik ke. CloudWatch

Untuk informasi selengkapnya, lihat Memeriksa Map Run.

Contoh definisi status Peta Terdistribusi

Gunakan `Map` status dalam mode Terdistribusi saat Anda perlu mengatur beban kerja paralel skala besar yang memenuhi kombinasi kondisi berikut:

- Ukuran dataset Anda melebihi 256 KB.

- Riwayat peristiwa eksekusi alur kerja melebihi 25.000 entri.
- Anda memerlukan konkurensi lebih dari 40 iterasi paralel.

Contoh definisi status Peta Terdistribusi berikut menentukan kumpulan data sebagai file CSV yang disimpan dalam bucket Amazon S3. Ini juga menentukan fungsi Lambda yang memproses data di setiap baris file CSV. Karena contoh ini menggunakan file CSV, itu juga menentukan lokasi header kolom CSV. Untuk melihat definisi mesin status lengkap dari contoh ini, lihat tutorial Menyalin data CSV skala besar menggunakan Peta Terdistribusi.

```
{
  "Map": {
    "Type": "Map",
    "ItemReader": {
      "ReaderConfig": {
        "InputType": "CSV",
        "CSVHeaderLocation": "FIRST_ROW"
      },
      "Resource": "arn:aws:states:::s3:getObject",
      "Parameters": {
        "Bucket": "Database",
        "Key": "csv-dataset/ratings.csv"
      }
    },
    "ItemProcessor": {
      "ProcessorConfig": {
        "Mode": "DISTRIBUTED",
        "ExecutionType": "EXPRESS"
      },
      "StartAt": "LambdaTask",
      "States": {
        "LambdaTask": {
          "Type": "Task",
          "Resource": "arn:aws:states:::lambda:invoke",
          "OutputPath": "$.Payload",
          "Parameters": {
            "Payload.$": "$",
            "FunctionName": "arn:aws:lambda:us-
east-2:123456789012:function:processCSVData"
          },
          "End": true
        }
      }
```

```
    },
    "Label": "Map",
    "End": true,
    "ResultWriter": {
      "Resource": "arn:aws:states:::s3:putObject",
      "Parameters": {
        "Bucket": "myOutputBucket",
        "Prefix": "csvProcessJobs"
      }
    }
  }
}
```

Izin untuk menjalankan Peta Terdistribusi

Bila Anda menyertakan status Peta Terdistribusi dalam alur kerja Anda, Step Functions memerlukan izin yang sesuai untuk memungkinkan peran mesin status menjalankan tindakan `StartExecution` API untuk status Peta Terdistribusi.

Contoh kebijakan IAM berikut memberikan hak istimewa paling sedikit yang diperlukan untuk peran mesin status Anda untuk menjalankan status Peta Terdistribusi.

> **Note**
>
> Pastikan Anda mengganti *stateMachineName* dengan nama mesin status tempat Anda menggunakan status Peta Terdistribusi. Misalnya, `arn:aws:states:`*us-east-2*`:`*123456789012*`:stateMachine:`*mystateMachine*.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution"
      ],
      "Resource": [
        "arn:aws:states:region:accountID:stateMachine:stateMachineName"
      ]
    },
    {
```

```
      "Effect": "Allow",
      "Action": [
        "states:DescribeExecution",
        "states:StopExecution"
      ],
      "Resource": "arn:aws:states:region:accountID:execution:stateMachineName:*"
    }
  ]
}
```

Selain itu, Anda perlu memastikan bahwa Anda memiliki hak istimewa paling sedikit yang diperlukan untuk mengakses AWS sumber daya yang digunakan dalam status Peta Terdistribusi, seperti bucket Amazon S3. Untuk informasi, lihat Kebijakan IAM untuk menggunakan status Peta Terdistribusi.

Bidang status Peta Terdistribusi

Untuk menggunakan status Peta Terdistribusi dalam alur kerja Anda, tentukan satu atau beberapa bidang ini. Anda menentukan bidang ini selain bidang status umum.

`Type` (Wajib)

Menetapkan jenis negara, sepertiMap.

`ItemProcessor` (Wajib)

Berisi objek JSON berikut yang menentukan mode pemrosesan `Map` status dan definisi.

- `ProcessorConfig`— Sebuah objek JSON yang menentukan konfigurasi untuk negara. `Map` Objek ini berisi sub-bidang berikut:
  - `Mode`— Setel **`DISTRIBUTED`** untuk menggunakan `Map` status dalam mode Terdistribusi.

    > **Note**
    >
    > Saat ini, jika Anda menggunakan `Map` status di dalam alur kerja Express, Anda tidak dapat mengatur `Mode` ke`DISTRIBUTED`. Namun, jika Anda menggunakan `Map` status di dalam alur kerja Standar, Anda dapat mengatur `Mode` ke`DISTRIBUTED`.

  - `ExecutionType`- Menentukan jenis eksekusi untuk alur kerja Peta sebagai STANDARD atau EXPRESS. Anda harus memberikan bidang ini jika Anda menentukan `DISTRIBUTED` untuk `Mode` sub-bidang. Untuk informasi selengkapnya tentang jenis alur kerja, lihatAlur Kerja Standar vs Ekspres.

- `StartAt`- Menentukan string yang menunjukkan keadaan pertama dalam alur kerja. String ini peka huruf besar/kecil dan harus cocok dengan nama salah satu objek state. Status ini berjalan pertama kali untuk setiap item dalam kumpulan data. Setiap input eksekusi yang Anda berikan ke `Map` status diteruskan ke `StartAt` status terlebih dahulu.
- `States`— Objek JSON yang berisi kumpulan status yang dibatasi koma. Dalam objek ini, Anda mendefinisikanMap workflow.

`ItemReader`

Menentukan dataset dan lokasinya. `Map`Negara menerima data inputnya dari kumpulan data yang ditentukan.

Dalam mode Terdistribusi, Anda dapat menggunakan payload JSON yang diteruskan dari status sebelumnya atau sumber data Amazon S3 skala besar sebagai kumpulan data. Untuk informasi selengkapnya, lihat ItemReader.

`ItemsPath` (Opsional)

Menentukan jalur referensi menggunakan JsonPathsintaks untuk memilih node JSON yang berisi array item di dalam input negara.

Dalam mode Terdistribusi, Anda menentukan bidang ini hanya ketika Anda menggunakan array JSON dari langkah sebelumnya sebagai input status Anda. Untuk informasi selengkapnya, lihat ItemsPath.

`ItemSelector` (Opsional)

Mengganti nilai item kumpulan data individu sebelum diteruskan ke setiap `Map` iterasi status.

Di bidang ini, Anda menentukan input JSON valid yang berisi kumpulan pasangan kunci-nilai. Pasangan ini dapat berupa nilai statis yang Anda tentukan dalam definisi mesin status Anda, nilai yang dipilih dari input status menggunakan jalur, atau nilai yang diakses dari objek konteks. Untuk informasi selengkapnya, lihat ItemSelector.

`ItemBatcher` (Opsional)

Menentukan untuk memproses item dataset dalam batch. Setiap eksekusi alur kerja anak kemudian menerima batch item ini sebagai input. Untuk informasi selengkapnya, lihat ItemBatcher.

`MaxConcurrency` (Opsional)

Menentukan jumlah eksekusi alur kerja anak yang dapat berjalan secara paralel. Penerjemah hanya mengizinkan hingga jumlah eksekusi alur kerja anak paralel yang ditentukan. Jika Anda

tidak menentukan nilai konkurensi atau menyetelnya ke nol, Step Functions tidak membatasi secara bersamaan dan menjalankan 10.000 eksekusi alur kerja anak paralel.

> **Note**
>
> Meskipun Anda dapat menentukan batas konkurensi yang lebih tinggi untuk eksekusi alur kerja anak paralel, sebaiknya Anda tidak melebihi kapasitas AWS layanan hilir, seperti. AWS Lambda

`MaxConcurrencyPath` (Opsional)

Jika Anda ingin memberikan nilai konkurensi maksimum secara dinamis dari input status menggunakan jalur referensi, gunakan. `MaxConcurrencyPath` Ketika diselesaikan, jalur referensi harus memilih bidang yang nilainya adalah bilangan bulat non-negatif.

> **Note**
>
> Suatu `Map` negara tidak dapat mencakup keduanya `MaxConcurrency` dan`MaxConcurrencyPath`.

`ToleratedFailurePercentage` (Opsional)

Mendefinisikan persentase item gagal untuk ditoleransi dalam Map Run. Map Run secara otomatis gagal jika melebihi persentase ini. Step Functions menghitung persentase item yang gagal sebagai hasil dari jumlah total item yang gagal atau habis waktu dibagi dengan jumlah item. Anda harus menentukan nilai antara nol dan 100. Untuk informasi selengkapnya, lihat Ambang kegagalan yang ditoleransi untuk status Peta Terdistribusi.

`ToleratedFailurePercentagePath` (Opsional)

Jika Anda ingin memberikan nilai persentase kegagalan yang ditoleransi secara dinamis dari input status menggunakan jalur referensi, gunakan. `ToleratedFailurePercentagePath` Ketika diselesaikan, jalur referensi harus memilih bidang yang nilainya antara nol dan 100.

`ToleratedFailureCount` (Opsional)

Mendefinisikan jumlah item gagal untuk ditoleransi dalam Map Run. Map Run secara otomatis gagal jika melebihi angka ini. Untuk informasi selengkapnya, lihat Ambang kegagalan yang ditoleransi untuk status Peta Terdistribusi.

`ToleratedFailureCountPath` (Opsional)

Jika Anda ingin memberikan nilai hitungan kegagalan yang ditoleransi secara dinamis dari input status menggunakan jalur referensi, gunakan. `ToleratedFailureCountPath` Ketika diselesaikan, jalur referensi harus memilih bidang yang nilainya adalah bilangan bulat non-negatif.

`Label` (Opsional)

String yang secara unik mengidentifikasi keadaan. `Map` Untuk setiap Map Run, Step Functions menambahkan label ke Map Run ARN. Berikut ini adalah contoh dari Map Run ARN dengan label kustom bernama: `demoLabel`

```
arn:aws:states:us-east-1:123456789012:mapRun:demoWorkflow/
demoLabel:3c39a231-69bb-3d89-8607-9e124eddbb0b
```

Jika Anda tidak menentukan label, Step Functions secara otomatis menghasilkan label unik.

> **Note**
>
> Label tidak boleh melebihi 40 karakter, harus unik dalam definisi mesin status, dan tidak dapat berisi salah satu karakter berikut:
>
> - Karakter spasi
> - Karakter wildcard (`? *`)
> - Karakter tanda kurung (`< > { } [ ]`)
> - Karakter khusus (`: ; , \ | ^ ~ $ # % & ` "`)
> - Karakter kontrol (`\\u0000` - `\\u001f` atau `\\u007f` - `\\u009f`).
>
> Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, aktivitas, dan label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

`ResultWriter` (Opsional)

Menentukan lokasi Amazon S3 tempat Step Functions menulis semua hasil eksekusi alur kerja anak.

Step Functions menggabungkan semua data eksekusi alur kerja anak, seperti input dan output eksekusi, ARN, dan status eksekusi. Kemudian mengekspor eksekusi dengan status yang sama

ke file masing-masing di lokasi Amazon S3 yang ditentukan. Untuk informasi selengkapnya, lihat ResultWriter.

Jika Anda tidak mengekspor hasil `Map` status, ia mengembalikan array dari semua hasil eksekusi alur kerja anak. Sebagai contoh:

```
[1, 2, 3, 4, 5]
```

`ResultPath` (Opsional)

Menentukan di mana dalam input untuk menempatkan output dari iterasi. Input kemudian disaring seperti yang ditentukan oleh `OutputPath`bidang jika ada, sebelum diteruskan sebagai output negara. Untuk informasi selengkapnya, lihat Pemrosesan Input dan Output.

`ResultSelector` (Opsional)

Lewati kumpulan pasangan kunci-nilai, di mana nilainya statis atau dipilih dari hasilnya. Untuk informasi selengkapnya, lihat ResultSelector.

> **Tip**
>
> Jika status Paralel atau Peta yang Anda gunakan di mesin status Anda mengembalikan array array, Anda dapat mengubahnya menjadi array datar dengan ResultSelector bidang. Untuk informasi selengkapnya, lihat Meratakan array array.

`Retry` (Opsional)

Array objek, yang disebut Retriers, yang mendefinisikan kebijakan coba lagi. Eksekusi menggunakan kebijakan coba lagi jika status mengalami error runtime. Untuk informasi selengkapnya, lihat Nyatakan contoh mesin menggunakan Coba Ulang dan menggunakan Catch.

> **Note**
>
> Jika Anda mendefinisikan Retrier untuk status Peta Terdistribusi, kebijakan coba lagi berlaku untuk semua eksekusi alur kerja anak yang dimulai status. `Map` Misalnya, bayangkan `Map` negara Anda memulai tiga eksekusi alur kerja anak, yang satu gagal. Ketika kegagalan terjadi, eksekusi menggunakan `Retry` bidang, jika ditentukan, untuk `Map` negara. Kebijakan coba lagi berlaku untuk semua eksekusi alur kerja anak dan bukan

hanya eksekusi yang gagal. Jika satu atau beberapa eksekusi alur kerja anak gagal, Map Run gagal.
Saat Anda mencoba lagi `Map` status, itu membuat Map Run baru.

`Catch` (Opsional)

Array objek, disebut Catch yang menentukan status fallback. Step Functions menggunakan Catchers yang didefinisikan `Catch` jika status mengalami kesalahan runtime. Ketika terjadi kesalahan, eksekusi pertama menggunakan retrier apa pun yang ditentukan dalam`Retry`. Jika kebijakan coba ulang tidak ditentukan atau habis, eksekusi menggunakan Catchers, jika ditentukan. Untuk informasi selengkapnya, lihat Status Fallback.

Langkah selanjutnya

Untuk terus mempelajari lebih lanjut tentang status Peta Terdistribusi, lihat sumber daya berikut:

- Pengolahan input dan output

  Untuk mengonfigurasi input yang diterima status Peta Terdistribusi dan output yang dihasilkannya, Step Functions menyediakan bidang berikut:

  - ItemReader
  - ItemsPath
  - ItemSelector
  - ItemBatcher
  - ResultWriter
  - Mengurai file CSV masukan

  Selain bidang ini, Step Functions juga memberi Anda kemampuan untuk menentukan ambang kegagalan yang ditoleransi untuk Peta Terdistribusi. Nilai ini memungkinkan Anda menentukan jumlah maksimum, atau persentase, item gagal sebagai ambang kegagalan untuk Map Run. Untuk informasi selengkapnya tentang mengonfigurasi ambang kegagalan yang ditoleransi, lihat. Ambang kegagalan yang ditoleransi untuk status Peta Terdistribusi
- Menggunakan status Peta Terdistribusi

  Lihat tutorial dan contoh proyek berikut untuk memulai menggunakan status Peta Terdistribusi.

  - Memulai dengan menggunakan status Peta Terdistribusi

- Memproses seluruh batch data dengan fungsi Lambda
- Memproses item data individual dengan fungsi Lambda
- Contoh proyek: Memproses file CSV dengan Peta Terdistribusi
- Contoh proyek: Memproses data dalam bucket Amazon S3 dengan Peta Terdistribusi

- Periksa eksekusi status Peta Terdistribusi

  Konsol Step Functions menyediakan halaman Map Run Details, yang menampilkan semua informasi yang terkait dengan eksekusi status Peta Terdistribusi. Untuk informasi tentang cara memeriksa informasi yang ditampilkan di halaman ini, lihatMemeriksa Map Run.

Ambang kegagalan yang ditoleransi untuk status Peta Terdistribusi

Saat Anda mengatur beban kerja paralel skala besar, Anda juga dapat menentukan ambang kegagalan yang ditoleransi. Nilai ini memungkinkan Anda menentukan jumlah maksimum, atau persentase, item gagal sebagai ambang kegagalan untuk Map Run. Bergantung pada nilai yang Anda tentukan, Map Run Anda gagal secara otomatis jika melebihi ambang batas. Jika Anda menentukan kedua nilai, alur kerja gagal ketika melebihi salah satu nilai.

Menentukan ambang batas membantu Anda gagal dalam jumlah item tertentu sebelum seluruh Map Run gagal. Step Functions mengembalikan `States.ExceedToleratedFailureThreshold` kesalahan ketika Map Run gagal karena ambang batas yang ditentukan terlampaui.

> **Note**
>
> Step Functions dapat terus menjalankan alur kerja turunan di Map Run bahkan setelah ambang kegagalan yang ditoleransi terlampaui, tetapi sebelum Map Run gagal.

Untuk menentukan nilai ambang batas di Workflow Studio, pilih Setel ambang kegagalan yang ditoleransi dalam Konfigurasi tambahan di bawah bidang Pengaturan waktu proses.

Persentase kegagalan yang ditoleransi

Mendefinisikan persentase item yang gagal untuk ditoleransi. Map Run Anda gagal jika nilai ini terlampaui. Step Functions menghitung persentase item yang gagal sebagai hasil dari jumlah total item yang gagal atau habis waktu dibagi dengan jumlah item. Anda harus menentukan nilai antara nol dan 100. Nilai persentase default adalah nol, yang berarti alur kerja gagal jika salah

satu eksekusi alur kerja anak gagal atau habis waktu. Jika Anda menentukan persentase sebagai 100, alur kerja tidak akan gagal meskipun semua eksekusi alur kerja anak gagal.

Atau, Anda dapat menentukan persentase sebagai jalur referensi ke pasangan kunci-nilai yang ada di masukan status Peta Terdistribusi Anda. Jalur ini harus menyelesaikan ke bilangan bulat positif antara 0 dan 100 saat runtime. Anda menentukan jalur referensi di `ToleratedFailurePercentagePath` sub-bidang.

Misalnya, diberikan input berikut:

```
{
  "percentage": 15
}
```

Anda dapat menentukan persentase menggunakan jalur referensi ke input tersebut sebagai berikut:

```
{
  ...
  "Map": {
    "Type": "Map",
    ...
    "ToleratedFailurePercentagePath": "$.percentage"
    ...
  }
}
```

> **Important**
>
> Anda dapat menentukan salah satu `ToleratedFailurePercentage` atau`ToleratedFailurePercentagePath`, tetapi tidak keduanya dalam definisi status Peta Terdistribusi Anda.

Jumlah kegagalan yang ditoleransi

Mendefinisikan jumlah item yang gagal untuk ditoleransi. Map Run Anda gagal jika nilai ini terlampaui.

Atau, Anda dapat menentukan hitungan sebagai jalur referensi ke pasangan kunci-nilai yang ada di masukan status Peta Terdistribusi Anda. Jalur ini harus menyelesaikan ke bilangan bulat positif saat runtime. Anda menentukan jalur referensi di `ToleratedFailureCountPath` sub-bidang.

Misalnya, diberikan input berikut:

```
{
  "count": 10
}
```

Anda dapat menentukan nomor menggunakan jalur referensi ke input tersebut sebagai berikut:

```
{
  ...
  "Map": {
    "Type": "Map",
    ...
    "ToleratedFailureCountPath": "$.count"
    ...
  }
}
```

> **Important**
>
> Anda dapat menentukan salah satu `ToleratedFailureCount` atau`ToleratedFailureCountPath`, tetapi tidak keduanya dalam definisi status Peta Terdistribusi Anda.

Transisi

Ketika Anda memulai eksekusi baru dari mesin negara Anda, sistem dimulai dengan status yang direferensikan di bidang tingkat atas`StartAt`. Bidang ini, diberikan sebagai string, harus sama persis, termasuk kasus, nama status dalam alur kerja.

Setelah status berjalan, AWS Step Functions gunakan nilai `Next` bidang untuk menentukan status berikutnya untuk maju ke.

`Next`bidang juga menentukan nama negara sebagai string. String ini peka huruf besar/kecil dan harus cocok dengan nama status yang ditentukan dalam deskripsi mesin negara dengan tepat

Misalnya, status berikut menyertakan transisi ke `NextState`.

```
"SomeState" : {
  ...,
  "Next" : "NextState"
}
```

Sebagian besar negara hanya mengizinkan satu aturan transisi dengan `Next` bidang tersebut. Namun, status kontrol aliran tertentu, seperti `Choice` status, memungkinkan Anda menentukan beberapa aturan transisi, masing-masing dengan bidangnya sendiri`Next`. Bahasa Status Amazon menyediakan detail tentang masing-masing tipe status yang dapat Anda tentukan, termasuk informasi tentang cara menentukan transisi.

Status dapat memiliki beberapa transisi masuk dari status lain.

Proses berulang sampai mencapai status terminal (status dengan`"Type": Succeed`,, atau`"End": true`)`"Type": Fail`, atau kesalahan runtime terjadi.

Ketika Anda redriveeksekusi, itu dianggap sebagai transisi status. Selain itu, semua status yang dijalankan kembali dalam a juga redrive dianggap sebagai transisi status.

Aturan berikut berlaku untuk status dalam mesin status:

- Negara dapat terjadi dalam urutan apa pun di dalam blok terlampir. Namun, urutan di mana mereka terdaftar tidak mempengaruhi urutan di mana mereka dijalankan. Perintah itu ditentukan oleh isi status.
- Dalam mesin negara, hanya ada satu negara yang ditunjuk sebagai `start` negara. `start`Status ditentukan oleh nilai `StartAt` bidang dalam struktur tingkat atas.
- Bergantung pada logika mesin status Anda — misalnya, jika mesin status Anda memiliki beberapa cabang logika — Anda mungkin memiliki lebih dari satu `end` status.
- Jika mesin negara Anda hanya terdiri dari satu status, itu bisa menjadi status awal dan akhir.

Transisi dalam status Peta Terdistribusi

Saat Anda menggunakan `Map` status dalam mode Terdistribusi, Anda akan dikenakan biaya satu transisi status untuk setiap eksekusi alur kerja anak yang memulai status Peta Terdistribusi. Saat Anda menggunakan `Map` status dalam mode Inline, Anda tidak dikenakan biaya transisi status untuk setiap iterasi status Peta Sebaris.

Anda dapat mengoptimalkan biaya dengan menggunakan `Map` status dalam mode Terdistribusi dan menyertakan alur kerja bersarang dalam definisi `Map` status. Status Peta Terdistribusi juga menambah nilai saat Anda memulai eksekusi alur kerja turunan dari tipe Express. Step Functions menyimpan respons dan status eksekusi alur kerja anak Express, yang mengurangi kebutuhan untuk menyimpan data eksekusi di CloudWatch Log. Anda juga bisa mendapatkan akses ke kontrol aliran yang tersedia dengan status Peta Terdistribusi, seperti menentukan ambang kesalahan atau mengelompokkan sekelompok item. Untuk informasi tentang harga Step Functions, lihat AWS Step Functionsharga.

Data Mesin Status

Data mesin status mengambil bentuk berikut:

- Input awal ke dalam mesin status
- Data yang diteruskan di antara status
- Output dari mesin status

Bagian ini menjelaskan cara data mesin status diformat dan digunakan dalam AWS Step Functions.

Topik

- Format Data
- Input/Output Mesin Status
- Input/output status

Format Data

Data mesin negara diwakili oleh teks JSON. Anda dapat memberikan nilai ke mesin negara menggunakan tipe data apa pun yang didukung oleh JSON.

> **Note**
>
> - Angka dalam format teks JSON sesuai dengan semantik. JavaScript Angka-angka ini biasanya sesuai dengan nilai IEEE-854 presisi ganda.
> - Berikut ini adalah teks JSON yang valid:
>   - String mandiri dan dibatasi kutipan

  - Objek
  - Array
  - Nomor
  - Nilai boolean
  - `null`
- Output dari negara menjadi masukan untuk negara berikutnya. Namun, Anda dapat membatasi status untuk bekerja pada subset data input dengan menggunakan Pemrosesan Input dan Output.

Input/Output Mesin Status

Anda dapat memberikan data input awal Anda ke mesin AWS Step Functions negara dengan salah satu dari dua cara. Anda dapat meneruskan data ke `StartExecution` tindakan ketika Anda memulai eksekusi. Anda juga dapat meneruskan data ke mesin status dari konsol Step Functions. Data awal diteruskan ke status `StartAt` mesin status. Jika input tidak tersedia, default-nya adalah obyek kosong (`{}`).

Output dari eksekusi dikembalikan oleh status terakhir (`terminal`). Output ini muncul sebagai teks JSON dalam hasil eksekusi ini.

Untuk Alur Kerja Standar, Anda dapat mengambil hasil eksekusi dari riwayat eksekusi menggunakan pemanggil eksternal, seperti tindakan. `DescribeExecution` Anda dapat melihat hasil eksekusi di konsol Step Functions.

Untuk Alur Kerja Ekspres, jika Anda mengaktifkan pencatatan, Anda dapat mengambil hasil dari CloudWatch Log, atau melihat dan men-debug eksekusi di konsol Step Functions. Untuk informasi selengkapnya, lihat Logging menggunakanCloudWatchLog dan Melihat dan men-debug eksekusi di konsol Step Functions.

Anda juga harus mempertimbangkan kuota yang terkait dengan mesin status Anda. Untuk informasi selengkapnya, lihat Kuota

Input/output status

Input setiap status terdiri dari teks JSON dari status sebelumnya atau, untuk status `StartAt`, input ke dalam eksekusi. Status kontrol aliran tertentu menggemakan input-nya ke output-nya.

Dalam contoh berikut, mesin status menambahkan dua angka bersama-sama.

1. Tentukan fungsi AWS Lambda.

```
function Add(input) {
  var numbers = JSON.parse(input).numbers;
  var total = numbers.reduce(
    function(previousValue, currentValue, index, array) {
      return previousValue + currentValue; });
  return JSON.stringify({ result: total });
}
```

2. Tentukan mesin status.

```
{
  "Comment": "An example that adds two numbers together.",
  "StartAt": "Add",
  "Version": "1.0",
  "TimeoutSeconds": 10,
  "States":
    {
        "Add": {
          "Type": "Task",
          "Resource": "arn:aws:lambda:us-east-1:123456789012:function:Add",
          "End": true
        }
    }
}
```

3. Mulai eksekusi dengan teks JSON berikut.

```
{ "numbers": [3, 4] }
```

   Status Add menerima teks JSON dan meneruskannya ke fungsi Lambda.

   Fungsi Lambda mengembalikan hasil perhitungan ke status tersebut.

   Status mengembalikan nilai berikut dalam output-nya.

```
{ "result": 7 }
```

Mengingat Add juga merupakan status akhir dalam mesin status, nilai ini dikembalikan sebagai output mesin status.

Jika status akhir tidak mengembalikan output, mesin status mengembalikan sebuah objek kosong (`{}`).

Untuk informasi selengkapnya, lihat Pengolahan Input dan Output di Step Functions.

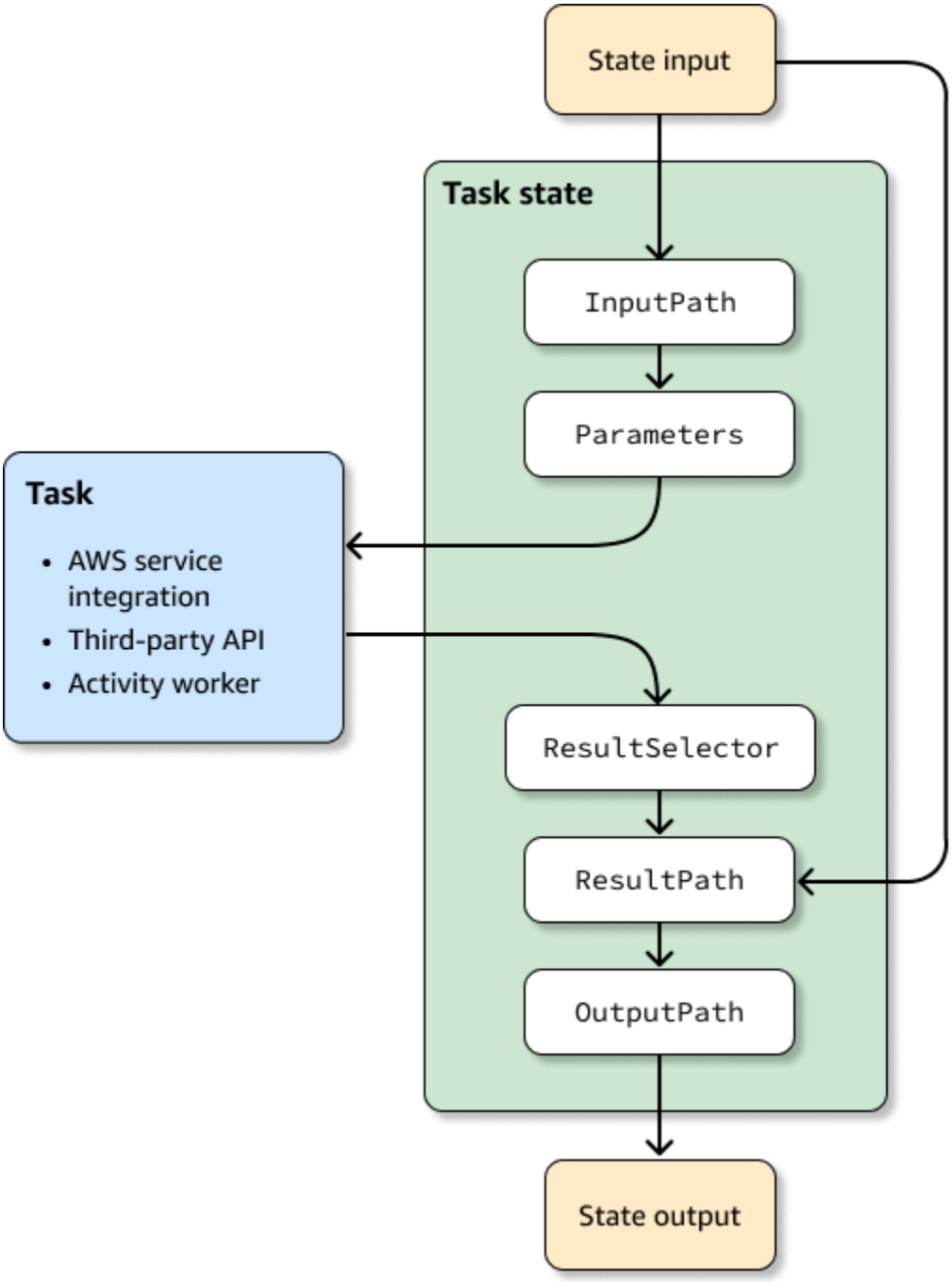
Pengolahan Input dan Output di Step Functions

Sebuah eksekusi Step Functions menerima teks JSON sebagai input dan meneruskan input yang ke status pertama dalam alur kerja. Status individu menerima JSON sebagai input dan biasanya lulus JSON sebagai output ke status berikutnya. Memahami cara informasi ini mengalir dari status ke status, dan belajar cara memfilter dan memanipulasi data ini, adalah kunci untuk secara efektif merancang dan menerapkan alur kerja di AWS Step Functions.

Dalam Bahasa Status Amazon, bidang ini memfilter dan mengontrol aliran JSON dari status ke status:

- `InputPath`
- `Parameters`
- `ResultSelector`
- `ResultPath`
- `OutputPath`

Diagram berikut menunjukkan bagaimana informasi JSON bergerak melalui status tugas. `InputPath`memilih bagian mana dari input JSON untuk diteruskan ke tugas `Task` negara (misalnya, AWS Lambda fungsi). `ResultPath`kemudian memilih kombinasi apa dari input status dan hasil tugas untuk diteruskan ke output. `OutputPath`dapat memfilter output JSON untuk lebih membatasi informasi yang diteruskan ke output.



`InputPath`, `Parameters`, `ResultSelector`, `ResultPath`, dan `OutputPath` masing-masing memanipulasi JSON saat bergerak melalui setiap status dalam alur kerja Anda.

Masing-masing dapat menggunakan jalur untuk memilih bagian dari JSON dari input atau hasilnya. Sebuah jalur adalah string, dimulai dengan $, yang mengidentifikasi simpul dalam teks JSON. Jalur Step Functions menggunakan JsonPathsintaks.

> **Tip**
>
> Gunakan simulator aliran data di konsol Step Functions untuk menguji sintaks jalur JSON, untuk lebih memahami bagaimana data dimanipulasi dalam status, dan untuk melihat bagaimana data dilewatkan antar status.

> **Tip**
>
> Untuk menerapkan contoh alur kerja yang menyertakan pemrosesan input dan output ke Anda Akun AWS, lihat Modul 6 - Pemrosesan Input dan Output dari Lokakarya. AWS Step Functions

Topik

- Jalan
- InputPath, Parameter dan ResultSelector
- ResultPath
- OutputPath
- InputPath,ResultPath, dan OutputPath Contoh
- Bidang input dan output negara peta
- Objek konteks

Jalan

Dalam Bahasa Status Amazon, jalur adalah string yang dimulai dengan $ yang dapat Anda gunakan untuk mengidentifikasi komponen dalam teks JSON. Jalur mengikuti JsonPathsintaks. Anda dapat menentukan jalur untuk mengakses himpunan bagian dari input ketika menentukan nilai untuk `InputPath`, `ResultPath`, dan `OutputPath`. Untuk informasi lebih lanjut, lihat Pengolahan Input dan Output di Step Functions.

> **Note**
>
> Anda juga dapat menentukan simpul JSON dari input atau objek konteks dengan menggunakan jalur dalam bidang `Parameters` dari ketentuan status. Lihat Meneruskan parameter ke API layanan.

Anda harus menggunakan notasi kurung siku jika nama bidang Anda berisi karakter apa pun yang tidak termasuk dalam `member-name-shorthand` definisi aturan JsonPath ABNF. Oleh karena itu, untuk menyandikan karakter khusus, seperti tanda baca (tidak termasuk_), Anda harus menggunakan notasi braket siku. Misalnya, `$.abc.['def ghi']`.

Jalur Referensi

Jalur referensi adalah jalur yang sintaksisterbatas sedemikian rupa sehingga dapat mengidentifikasi hanya satu simpul dalam struktur JSON:

- Anda dapat mengakses bidang objek hanya menggunakan notasi titik (`.`) dan kurung persegi (`[ ]`).
- Fungsi seperti `length()` tidak didukung.
- Operator leksikal, yang non-simbolis, seperti `subsetof` tidak didukung.
- Pemfilteran berdasarkan ekspresi reguler atau dengan mereferensikan nilai lain dalam struktur JSON tidak didukung.
- @Operator, yang cocok dengan node saat ini yang sedang diproses dalam filter, tidak cocok dengan nilai skalar. Itu hanya cocok dengan objek.

Misalnya, jika data input status berisi nilai berikut:

```
{
  "foo": 123,
  "bar": ["a", "b", "c"],
  "car": {
      "cdr": true
  },
  "jar": [{"a": 1}, {"a": 5}, {"a": 2}, {"a": 7}, {"a": 3}]
}
```

Jalur referensi berikut akan mengembalikan berikut.

```
$.foo => 123
$.bar => ["a", "b", "c"]
$.car.cdr => true
$.jar[?(@.a >= 5)] => [{"a": 5}, {"a": 7}]
```

Status tertentu menggunakan jalur dan jalur referensi untuk mengontrol aliran mesin status atau mengonfigurasi pengaturan status atau opsi. Untuk informasi selengkapnya, lihat Memodelkan pemrosesan jalur input dan output alur kerja dengan simulator aliran data dan Menggunakan JsonPath secara efektif di. AWS Step Functions

Meratakan array array

Jika Map status Paralel atau di mesin status Anda mengembalikan array array, Anda dapat mengubahnya menjadi array datar dengan ResultSelector bidang. Anda dapat menyertakan bidang ini di dalam definisi status Paralel atau Peta untuk memanipulasi hasil status ini.

Untuk meratakan array, gunakan sintaks JMESPath `[*]` di `ResultSelector` bidang seperti yang ditunjukkan pada contoh berikut.

```
"ResultSelector": {
    "flattenArray.$": "$[*][*]"
  }
```

Untuk contoh yang menunjukkan cara meratakan array, lihat Langkah 3 dalam tutorial berikut:

- Memproses seluruh batch data dengan fungsi Lambda
- Memproses item data individual dengan fungsi Lambda

InputPath, Parameter dan ResultSelector

Bidang `InputPath`, `Parameters` dan `ResultSelector` menyediakan cara untuk memanipulasi JSON saat bergerak melalui alur kerja Anda. `InputPath` dapat membatasi input yang diteruskan dengan memfilter notasi JSON dengan menggunakan jalur (lihat Jalan). Bidang `Parameters` mengaktifkan Anda untuk meneruskan kumpulan pasangan kunci-nilai, yang mana nilai-nilai yang baik nilai-nilai statis yang Anda tetapkan dalam ketetapan mesin negara Anda, atau yang dipilih dari input menggunakan jalur. Bidang `ResultSelector` menyediakan cara untuk memanipulasi hasil status sebelum `ResultPath` diterapkan.

AWS Step Functions menerapkan `InputPath` bidang terlebih dahulu, dan kemudian `Parameters` bidang. Pertama-tama Anda dapat memfilter input mentah Anda ke pilihan yang ingin Anda gunakan `InputPath`, lalu menerapkan `Parameters` untuk memanipulasi input itu lebih lanjut, atau menambahkan nilai-nilai baru. Kemudian Anda dapat menggunakan bidang `ResultSelector` untuk memanipulasi output status sebelum `ResultPath` diterapkan.

> **Tip**
>
> Gunakan simulator aliran data di konsol Step Functions untuk menguji sintaks jalur JSON, untuk lebih memahami bagaimana data dimanipulasi dalam status, dan untuk melihat bagaimana data dilewatkan antar status.

InputPath

Gunakan `InputPath` untuk memilih sebagian dari input status.

Misalnya, anggap input ke status Anda meliputi berikut ini.

```
{
  "comment": "Example for InputPath.",
  "dataset1": {
    "val1": 1,
    "val2": 2,
    "val3": 3
  },
  "dataset2": {
    "val1": "a",
    "val2": "b",
    "val3": "c"
  }
}
```

Anda dapat menerapkan `InputPath`.

```
"InputPath": "$.dataset2",
```

Dengan `InputPath` sebelumnya, berikut ini adalah JSON yang diteruskan sebagai input.

```
{
```

```
  "val1": "a",
  "val2": "b",
  "val3": "c"
}
```

> **Note**
>
> Sebuah jalur dapat menghasilkan pilihan nilai. Pertimbangkan contoh berikut.
>
> ```
> { "a": [1, 2, 3, 4] }
> ```
>
> Jika Anda menerapkan jalur `$.a[0:2]`, berikut ini adalah hasilnya.
>
> ```
> [ 1, 2 ]
> ```

Parameter

Bagian ini menjelaskan berbagai cara Anda dapat menggunakan bidang Parameter.

Pasangan kunci/nilai

Gunakan bidang `Parameters` untuk membuat kumpulan pasangan kunci-nilai yang diteruskan sebagai input. Nilai masing-masing dapat menjadi berupa nilai-nilai statis yang Anda masukkan dalam ketentuan mesin status Anda, atau dipilih baik dari input atau objek konteks dengan jalur. Untuk pasangan kunci-nilai tempat nilai yang dipilih menggunakan jalur, nama kunci harus diakhiri `.$`.

Misalnya, anggap Anda memberikan input berikut.

```
{
  "comment": "Example for Parameters.",
  "product": {
    "details": {
       "color": "blue",
       "size": "small",
       "material": "cotton"
    },
    "availability": "in stock",
    "sku": "2317",
```

```
    "cost": "$23"
  }
}
```

Untuk memilih beberapa informasi, Anda dapat menentukan parameter ini dalam ketentuan mesin status Anda.

```
"Parameters": {
        "comment": "Selecting what I care about.",
        "MyDetails": {
          "size.$": "$.product.details.size",
          "exists.$": "$.product.availability",
          "StaticValue": "foo"
        }
      },
```

Mengingat input dan bidang `Parameters` sebelumnya, ini adalah JSON yang dilewatkan.

```
{
  "comment": "Selecting what I care about.",
  "MyDetails": {
      "size": "small",
      "exists": "in stock",
      "StaticValue": "foo"
  }
},
```

Selain input, Anda dapat mengakses objek JSON khusus, yang dikenal sebagai objek konteks. Objek konteks mencakup informasi tentang eksekusi mesin status Anda. Lihat Objek konteks.

Sumber daya terhubung

Bidang `Parameters` juga dapat meneruskan informasi ke sumber daya yang terhubung. Misalnya, jika status tugas Anda mengatur AWS Batch pekerjaan, Anda dapat meneruskan parameter API yang relevan secara langsung ke tindakan API layanan tersebut. Lihat informasi yang lebih lengkap di:

- Meneruskan parameter ke API layanan
- Bekerja dengan layanan yang lain

Amazon S3

Jika data fungsi Lambda yang Anda lewati antar status mungkin bertambah menjadi lebih dari 262.144 byte, sebaiknya gunakan Amazon S3 untuk menyimpan data, dan menerapkan salah satu metode berikut:

- Gunakan status Peta Terdistribusi di alur kerja Anda sehingga `Map` status dapat membaca input langsung dari sumber data Amazon S3. Untuk informasi selengkapnya, lihat Menggunakan status Peta dalam mode Terdistribusi.
- Parse Amazon Resource Name (ARN) bucket dalam `Payload` parameter untuk mendapatkan nama bucket dan nilai kunci. Untuk informasi selengkapnya, lihat Gunakan ARN Amazon S3 bukan meneruskan muatan besar.

Atau, Anda dapat menyesuaikan implementasi untuk meneruskan muatan yang lebih kecil dalam eksekusi Anda.

ResultSelector

Gunakan bidang `ResultSelector` untuk memanipulasi hasil status sebelum `ResultPath` diterapkan. Bidang `ResultSelector` memungkinkan Anda membuat koleksi pasangan nilai kunci, yang mana nilai-nilainya statis atau dipilih dari hasil status. Dengan menggunakan `ResultSelector` bidang, Anda dapat memilih bagian mana dari hasil status yang ingin Anda lewatkan ke `ResultPath` bidang tersebut.

> **Note**
>
> Dengan `ResultPath` bidang tersebut, Anda dapat menambahkan output `ResultSelector` bidang ke input asli.

`ResultSelector` adalah bidang opsional dalam status berikut:

- Map
- Tugas
- Paralel

Misalnya, integrasi layanan Step Functions mengembalikan metadata selain muatan dalam hasil. `ResultSelector` dapat memilih bagian dari hasil dan menggabungkannya dengan input status

dengan `ResultPath`. Dalam contoh ini, kami ingin memilih hanya `resourceType` dan `ClusterId`, dan menggabungkannya dengan input status dari Amazon EMR createCluster.sync. Diberikan sebagai berikut:

```
{
  "resourceType": "elasticmapreduce",
  "resource": "createCluster.sync",
  "output": {
    "SdkHttpMetadata": {
      "HttpHeaders": {
        "Content-Length": "1112",
        "Content-Type": "application/x-amz-JSON-1.1",
        "Date": "Mon, 25 Nov 2019 19:41:29 GMT",
        "x-amzn-RequestId": "1234-5678-9012"
      },
      "HttpStatusCode": 200
    },
    "SdkResponseMetadata": {
      "RequestId": "1234-5678-9012"
    },
    "ClusterId": "AKIAIOSFODNN7EXAMPLE"
  }
}
```

Anda kemudian dapat memilih `resourceType` dan `ClusterId` menggunakan `ResultSelector`:

```
"Create Cluster": {
  "Type": "Task",
  "Resource": "arn:aws:states:::elasticmapreduce:createCluster.sync",
  "Parameters": {
    <some parameters>
  },
  "ResultSelector": {
    "ClusterId.$": "$.output.ClusterId",
    "ResourceType.$": "$.resourceType"
  },
  "ResultPath": "$.EMROutput",
  "Next": "Next Step"
}
```

Dengan input yang diberikan, menggunakan `ResultSelector` menghasilkan:

```
{
  "OtherDataFromInput": {},
  "EMROutput": {
    "ResourceType": "elasticmapreduce",
    "ClusterId": "AKIAIOSFODNN7EXAMPLE"
  }
}
```

Meratakan array array

Jika Map status Paralel atau di mesin status Anda mengembalikan array array, Anda dapat mengubahnya menjadi array datar dengan ResultSelector bidang. Anda dapat menyertakan bidang ini di dalam definisi status Paralel atau Peta untuk memanipulasi hasil status ini.

Untuk meratakan array, gunakan sintaks JMESPath `[*]` di `ResultSelector` bidang seperti yang ditunjukkan pada contoh berikut.

```
"ResultSelector": {
    "flattenArray.$": "$[*][*]"
  }
```

Untuk contoh yang menunjukkan cara meratakan array, lihat Langkah 3 dalam tutorial berikut:

- Memproses seluruh batch data dengan fungsi Lambda
- Memproses item data individual dengan fungsi Lambda

ResultPath

Output dari suatu status dapat menjadi salinan inputnya, hasilnya menghasilkan (misalnya, output dari fungsi Lambda status `Task`), atau kombinasi dari input dan hasilnya. Gunakan `ResultPath` untuk kombinasi mana dari ini yang diteruskan ke output status.

Tipe negara berikut dapat menghasilkan hasil dan dapat mencakup `ResultPath`:

- Diteruskan
- Tugas
- Paralel
- Map

Gunakan `ResultPath` untuk menggabungkan hasil tugas dengan input tugas, atau untuk memilih salah satu dari ini. Jalan yang Anda berikan untuk `ResultPath` mengontrol informasi apa yang diteruskan ke output.

> **Note**
>
> `ResultPath` terbatas untuk menggunakan jalur referensi, yang membatasi ruang lingkup sehingga dapat mengidentifikasi hanya satu simpul di JSON. Lihat Jalur Referensi di Bahasa Status Amazon.

Contoh-contoh ini didasarkan pada mesin status dan fungsi Lambda yang dijelaskan dalam tutorial Membuat mesin status Step Functions yang menggunakan Lambda. Bekerja melalui tutorial itu dan menguji output yang berbeda dengan mencoba berbagai jalur dalam bidang `ResultPath`.

Gunakan ResultPath untuk:

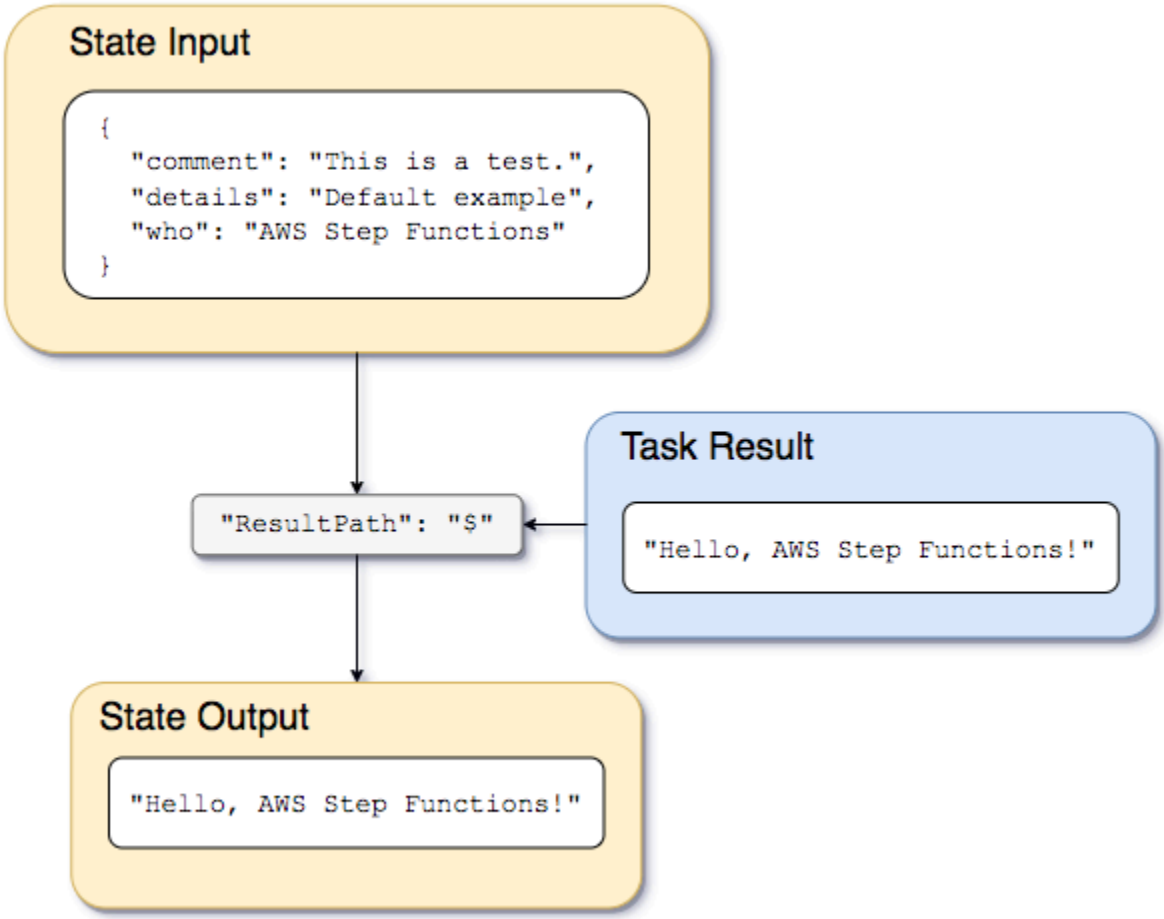
- Gunakan ResultPath untuk Mengganti Input dengan Hasil
- Buang Hasil dan Simpan Input Asli
- Gunakan ResultPath untuk Menyertakan Hasil dengan Input
- Gunakan ResultPath untuk Memperbarui Node di Input dengan Hasil
- Gunakan ResultPath untuk Menyertakan Kesalahan dan Input dalam Catch

> **Tip**
>
> Gunakan simulator aliran data di konsol Step Functions untuk menguji sintaks jalur JSON, untuk lebih memahami bagaimana data dimanipulasi dalam status, dan untuk melihat bagaimana data dilewatkan antar status.

Gunakan ResultPath untuk Mengganti Input dengan Hasil

Jika Anda tidak menentukan `ResultPath`, perilaku default adalah seolah-olah Anda telah menentukan `"ResultPath": "$"`. Karena hal ini memberitahu status untuk mengganti seluruh input dengan hasil, input status benar-benar digantikan oleh hasil yang berasal dari hasil tugas.

Diagram berikut menunjukkan bagaimana `ResultPath` benar-benar dapat mengganti input dengan hasil tugas.



Gunakan mesin status dan fungsi Lambda yang dijelaskan dalamMembuat mesin status Step Functions yang menggunakan Lambda, dan ubah jenis integrasi layanan menjadi integrasi AWS SDK untuk fungsi Lambda. Untuk melakukan ini, tentukan fungsi Lambda Amazon Resource Name (ARN) di `Resource` bidang `Task` status seperti yang ditunjukkan pada contoh berikut. Menggunakan integrasi AWS SDK memastikan bahwa hasil `Task` status hanya berisi output fungsi Lambda tanpa metadata apa pun.

```
{
   "StartAt":"CallFunction",
   "States":{
      "CallFunction": {
         "Type":"Task",
         "Resource":"arn:aws:lambda:us-east-2:123456789012:function:HelloFunction",
         "End": true
      }
   }
```

```
}
```

Kemudian, berikan masukan berikut:

```
{
  "comment": "This is a test of the input and output of a Task state.",
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

Fungsi Lambda memberikan hasil sebagai berikut.

```
"Hello, AWS Step Functions!"
```

> **Tip**
>
> Anda dapat melihat hasil ini di Step Functionskonsol. Untuk melakukan ini, pada halaman Rincian Eksekusi konsol, pilih Lambda fungsi dalam tampilan Grafik. Kemudian, pilih Output tab di Detail langkah panel untuk melihat hasil ini.

Jika `ResultPath` tidak dispesifikasikan di status, atau jika `"ResultPath": "$"` diatur, input status digantikan oleh hasil fungsi Lambda, dan output status adalah sebagai berikut.

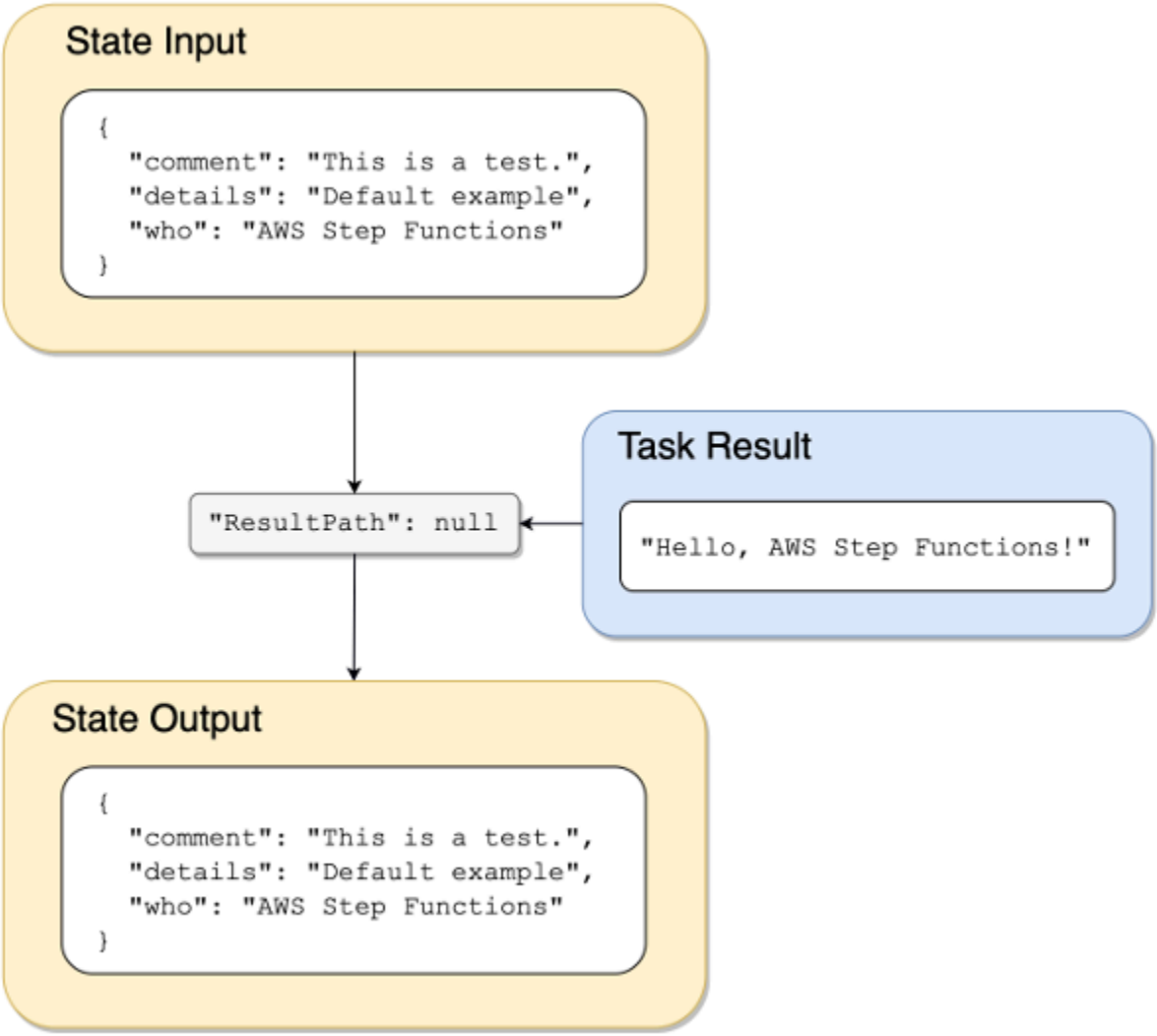
```
"Hello, AWS Step Functions!"
```

> **Note**
>
> `ResultPath` digunakan untuk memasukkan konten dari hasil dengan input, sebelum meneruskannya ke output. Tapi, jika `ResultPath` tidak ditentukan, default-nya adalah mengganti seluruh input.

Buang Hasil dan Simpan Input Asli

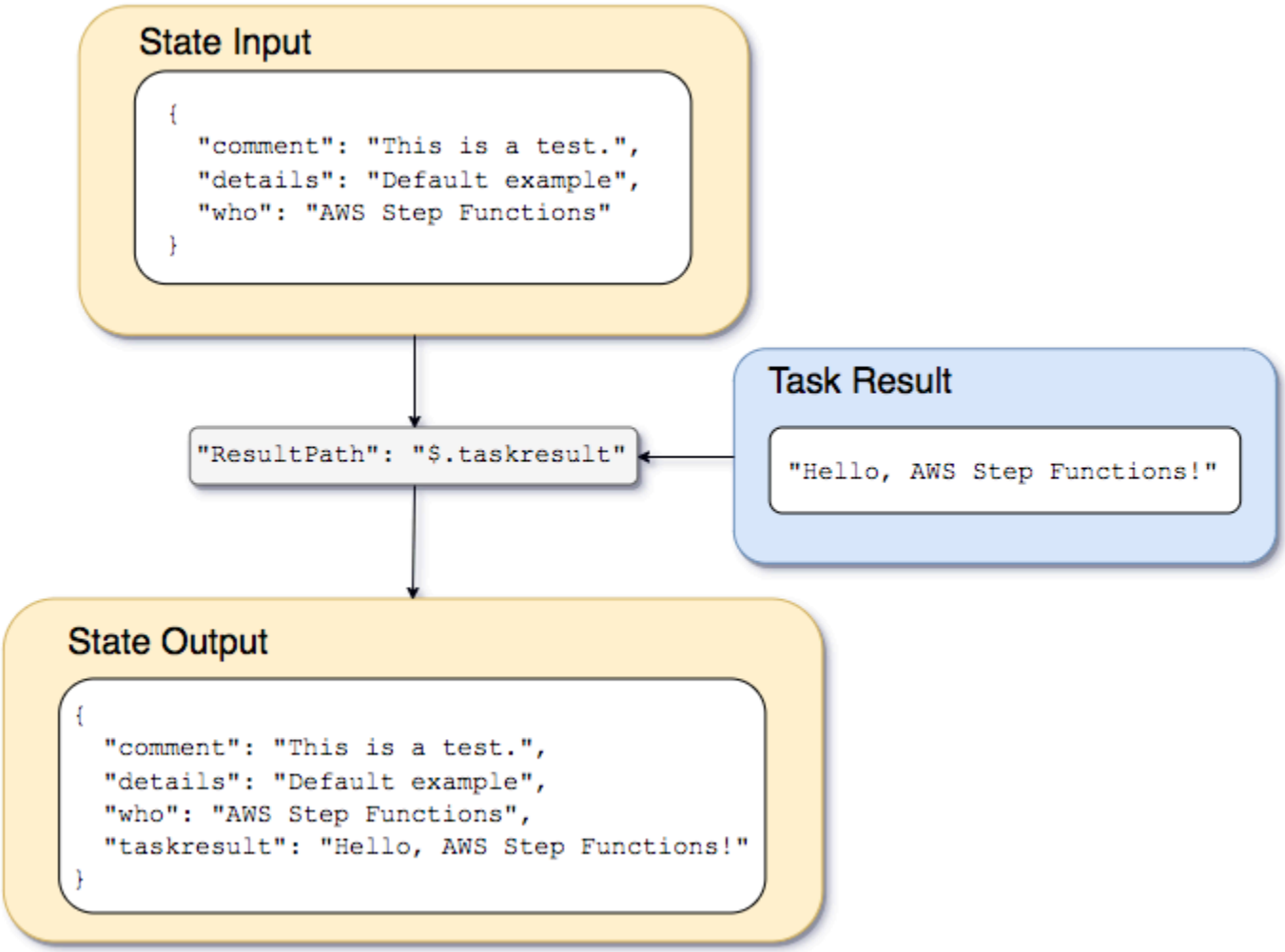
Jika Anda mengatur `ResultPath` ke `null`, itu akan meneruskan input asli ke output. Menggunakan `"ResultPath": null`, muatan input status akan disalin langsung ke output, tanpa memperhatikan hasilnya.

Diagram berikut menunjukkan bagaimana `ResultPath` null akan menyalin input langsung ke output.



Gunakan ResultPath untuk Menyertakan Hasil dengan Input

Diagram berikut menunjukkan bagaimana `ResultPath` dapat menyertakan hasilnya dengan input.



Menggunakan mesin status dan fungsi Lambda dijelaskan dalam tutorial Membuat mesin status Step Functions yang menggunakan Lambda, kita bisa meneruskan input berikut.

```
{
  "comment": "This is a test of the input and output of a Task state.",
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

Hasil dari fungsi Lambda adalah sebagai berikut.

```
"Hello, AWS Step Functions!"
```

Untuk mempertahankan input, masukkan hasil dari fungsi Lambda, lalu teruskan JSON gabungan ke status berikutnya, kita bisa mengatur `ResultPath` ke hal berikut.

```
"ResultPath": "$.taskresult"
```

Ini termasuk hasil dari fungsi Lambda dengan input asli.

```
{
  "comment": "This is a test of input and output of a Task state.",
  "details": "Default behavior example",
  "who": "AWS Step Functions",
  "taskresult": "Hello, AWS Step Functions!"
}
```

Output dari fungsi Lambda ditambahkan ke input asli sebagai nilai untuk `taskresult`. Input, termasuk nilai yang baru dimasukkan, diteruskan ke status berikutnya.

Anda juga dapat memasukkan hasilnya ke simpul anak dari input. Atur `ResultPath` ke hal berikut.

```
"ResultPath": "$.strings.lambdaresult"
```

Mulai eksekusi menggunakan input berikut.

```
{
  "comment": "An input comment.",
  "strings": {
    "string1": "foo",
    "string2": "bar",
    "string3": "baz"
  },
  "who": "AWS Step Functions"
}
```

Hasil dari fungsi Lambda dimasukkan sebagai anak dari simpul `strings` dalam input.

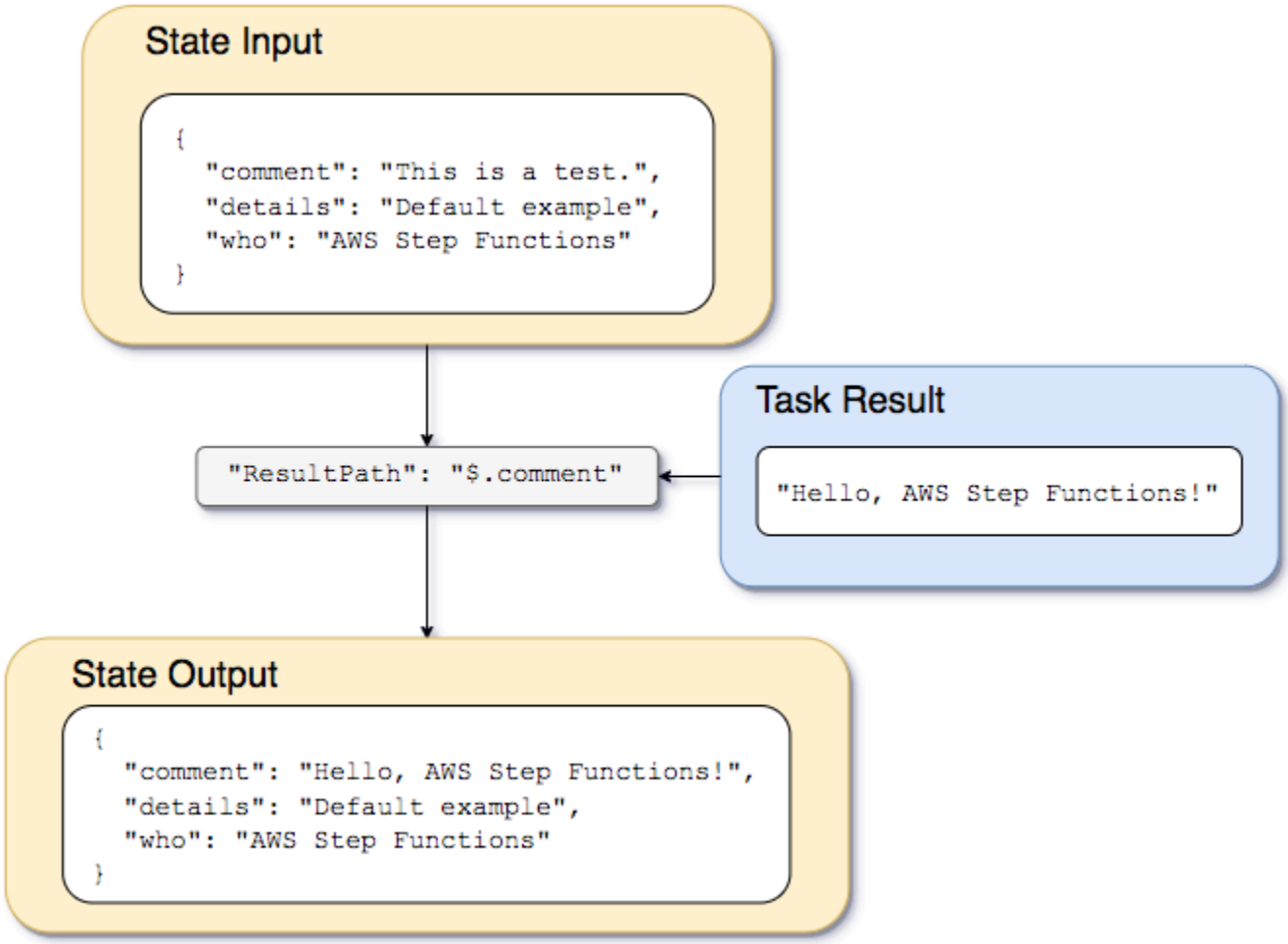
```
{
  "comment": "An input comment.",
  "strings": {
    "string1": "foo",
    "string2": "bar",
    "string3": "baz",
```

```
    "lambdaresult": "Hello, AWS Step Functions!"
  },
  "who": "AWS Step Functions"
}
```

Output status sekarang termasuk input asli JSON dengan hasil sebagai simpul anak.

Gunakan ResultPath untuk Memperbarui Node di Input dengan Hasil

Diagram berikut menunjukkan bagaimana `ResultPath` dapat memperbarui nilai simpul JSON yang ada di input dengan nilai-nilai dari hasil tugas.



Menggunakan contoh mesin status dan fungsi Lambda yang dijelaskan dalam tutorial Membuat mesin status Step Functions yang menggunakan Lambda, kita bisa meneruskan input berikut.

```
{
```

```
  "comment": "This is a test of the input and output of a Task state.",
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

Hasil dari fungsi Lambda adalah sebagai berikut.

```
Hello, AWS Step Functions!
```

Alih-alih mempertahankan input dan memasukkan hasilnya sebagai simpul baru di JSON, kita dapat menimpa simpul yang ada.

Sebagai contoh, sama seperti menghilangkan atau menetapkan `"ResultPath": "$"` yang menimpa seluruh simpul, Anda dapat menentukan simpul individu untuk menimpa dengan hasilnya.

```
"ResultPath": "$.comment"
```

Karena simpul `comment` sudah ada di input status, mengatur `ResultPath` ke `"$.comment"` menggantikan simpul di input dengan hasil dari fungsi Lambda. Tanpa pemfilteran lebih lanjut oleh `OutputPath`, berikut ini diteruskan ke output.

```
{
  "comment": "Hello, AWS Step Functions!",
  "details": "Default behavior example",
  "who": "AWS Step Functions",
}
```

Nilai untuk simpul `comment`, `"This is a test of the input and output of a Task state."`, digantikan oleh hasil fungsi Lambda: `"Hello, AWS Step Functions!"` dalam output status.

Gunakan ResultPath untuk Menyertakan Kesalahan dan Input dalam **`Catch`**

Tutorial Menangani kondisi kesalahan menggunakan mesin status Step Functions ini menunjukkan cara menggunakan mesin status untuk menangkap kesalahan. Dalam beberapa kasus, Anda mungkin ingin mempertahankan input asli dengan kesalahan. Gunakan `ResultPath` dalam `Catch` untuk memasukkan kesalahan dengan input asli, bukan menggantikannya.

```
"Catch": [{
```

```
  "ErrorEquals": ["States.ALL"],
  "Next": "NextTask",
  "ResultPath": "$.error"
}]
```

Jika pernyataan `Catch` sebelumnya menangkap kesalahan, pernyataan itu mencakup hasil dalam simpul `error` dalam input status. Sebagai contoh, dengan input berikut:

```
{"foo": "bar"}
```

Output status saat menangkap kesalahan adalah sebagai berikut.

```
{
  "foo": "bar",
  "error": {
    "Error": "Error here"
  }
}
```

Untuk informasi selengkapnya tentang penanganan kesalahan, lihat hal berikut:

- Penanganan kesalahan dalam Step Functions
- Menangani kondisi kesalahan menggunakan mesin status Step Functions

OutputPath

`OutputPath` mengaktifkan Anda untuk memilih sebagian dari output status untuk meneruskan ke status berikutnya. Hal ini mengaktifkan Anda untuk memfilter informasi yang tidak diinginkan, dan meneruskan hanya bagian dari JSON yang Anda perlukan.

Jika Anda tidak menentukan `OutputPath` nilai default-nya adalah `$`. Ini meneruskan seluruh simpul JSON (ditentukan oleh input status, hasil tugas, dan `ResultPath`) ke status berikutnya.

> **Tip**
>
> Gunakan simulator aliran data di konsol Step Functions untuk menguji sintaks jalur JSON, untuk lebih memahami bagaimana data dimanipulasi dalam status, dan untuk melihat bagaimana data dilewatkan antar status.

Untuk informasi selengkapnya, lihat hal berikut:

- Jalur dalam Bahasa Amazon States
- InputPath,ResultPath, dan OutputPath Contoh
- Lulus JSON statis sebagai parameter
- Pengolahan Input dan Output di Step Functions

InputPath,ResultPath, dan OutputPath Contoh

Setiap negara selain Gagal negara atau Berhasil negara dapat mencakup input dan output bidang pengolahan, sepertiInputPath,ResultPath, atauOutputPath. Selain itu, Tunggu dan Pilihan negara tidak mendukung ResultPath bidang. Dengan bidang ini, Anda dapat menggunakan JsonPathuntuk memfilter data JSON saat bergerak melalui alur kerja Anda.

Anda juga dapat menggunakan Parameters bidang untuk memanipulasi data JSON saat bergerak melalui alur kerja Anda. Untuk informasi tentang penggunaanParameters, lihatInputPath, Parameter dan ResultSelector.

Sebagai contoh, mulai dengan fungsi dan mesin status AWS Lambda yang dijelaskan dalam tutorial Membuat mesin status Step Functions yang menggunakan Lambda. Modifikasi mesin status sehingga termasuk InputPath, ResultPath, dan OutputPath berikut.

```
{
  "Comment": "A Hello World example of the Amazon States Language using an AWS Lambda
 function",
  "StartAt": "HelloWorld",
  "States": {
    "HelloWorld": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:HelloFunction",
      "InputPath": "$.lambda",
      "ResultPath": "$.data.lambdaresult",
      "OutputPath": "$.data",
      "End": true
    }
  }
}
```

Mulai eksekusi menggunakan input berikut.

```
{
  "comment": "An input comment.",
  "data": {
    "val1": 23,
    "val2": 17
  },
  "extra": "foo",
  "lambda": {
    "who": "AWS Step Functions"
  }
}
```

Asumsikan bahwa simpul `comment` dan `extra` dapat dibuang, tetapi bahwa kita ingin menyertakan output dari fungsi Lambda, dan mempertahankan informasi dalam simpul `data`.

Dalam mesin status yang diperbarui, status `Task` diubah untuk memproses input ke tugas.

```
"InputPath": "$.lambda",
```

Baris ini dalam ketentuan mesin status membatasi input tugas untuk hanya simpul `lambda` dari input status. Fungsi Lambda hanya menerima objek JSON `{"who": "AWS Step Functions"}` sebagai input.

```
"ResultPath": "$.data.lambdaresult",
```

Ini `ResultPath` memberitahu mesin status untuk memasukkan hasil dari fungsi Lambda ke simpul bernama `lambdaresult`, sebagai anak dari simpul `data` dalam input mesin status asli. Karena kita tidak melakukan manipulasi lain pada input asli dan hasilnya menggunakan`OutputPath`, output dari state sekarang termasuk hasil dari fungsi Lambda dengan input asli.

```
{
  "comment": "An input comment.",
  "data": {
    "val1": 23,
    "val2": 17,
    "lambdaresult": "Hello, AWS Step Functions!"
  },
  "extra": "foo",
  "lambda": {
    "who": "AWS Step Functions"
```

```
  }
}
```

Tapi, tujuan kami adalah untuk hanya mempertahankan simpul `data`, dan termasuk hasil fungsi Lambda. `OutputPath` memfilter JSON gabungan ini sebelum meneruskannya ke output status.

```
"OutputPath": "$.data",
```

Ini hanya memilih simpul `data` dari input asli (termasuk anak `lambdaresult` yang disisipkan oleh `ResultPath`) untuk diteruskan ke output. Output status difilter ke berikut ini.

```
{
  "val1": 23,
  "val2": 17,
  "lambdaresult": "Hello, AWS Step Functions!"
}
```

Dalam status `Task` ini:

1. `InputPath` hanya mengirimkan simpul `lambda` dari input ke fungsi Lambda.
2. `ResultPath` menyisipkan hasilnya sebagai anak dari simpul `data` dalam input asli.
3. `OutputPath` memfilter input status (yang sekarang termasuk hasil dari fungsi Lambda) sehingga hanya meneruskan simpul `data` ke output status.

Example untuk memanipulasi input mesin negara asli, hasil, dan output akhir menggunakan JsonPath

Pertimbangkan mesin negara berikut yang memverifikasi identitas dan alamat pemohon asuransi.

> **Note**
>
> Untuk melihat contoh lengkapnya, lihat Cara menggunakan JSON Path in Step Functions.

```
{
  "Comment": "Sample state machine to verify an applicant's ID and address",
  "StartAt": "Verify info",
  "States": {
    "Verify info": {
      "Type": "Parallel",
```

```
      "End": true,
      "Branches": [
        {
          "StartAt": "Verify identity",
          "States": {
            "Verify identity": {
              "Type": "Task",
              "Resource": "arn:aws:states:::lambda:invoke",
              "Parameters": {
                "Payload.$": "$",
                "FunctionName": "arn:aws:lambda:us-east-2:111122223333:function:check-
identity:$LATEST"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Verify address",
          "States": {
            "Verify address": {
              "Type": "Task",
              "Resource": "arn:aws:states:::lambda:invoke",
              "Parameters": {
                "Payload.$": "$",
                "FunctionName": "arn:aws:lambda:us-east-2:111122223333:function:check-
address:$LATEST"
              },
              "End": true
            }
          }
        }
      ]
    }
  }
}
```

Jika Anda menjalankan mesin status ini menggunakan input berikut, eksekusi gagal karena fungsi Lambda yang melakukan verifikasi hanya mengharapkan data yang perlu diverifikasi sebagai input. Oleh karena itu, Anda harus menentukan node yang berisi informasi yang akan diverifikasi menggunakan yang sesuaiJsonPath.

```
{
```

```
  "data": {
    "firstname": "Jane",
    "lastname": "Doe",
    "identity": {
      "email": "jdoe@example.com",
      "ssn": "123-45-6789"
    },
    "address": {
      "street": "123 Main St",
      "city": "Columbus",
      "state": "OH",
      "zip": "43219"
    },
    "interests": [
      {
        "category": "home",
        "type": "own",
        "yearBuilt": 2004
      },
      {
        "category": "boat",
        "type": "snowmobile",
        "yearBuilt": 2020
      },
      {
        "category": "auto",
        "type": "RV",
        "yearBuilt": 2015
      },
    ]
  }
}
```

Untuk menentukan node yang harus digunakan fungsi *`check-identity`* Lambda, gunakan `InputPath` bidang sebagai berikut:

```
"InputPath": "$.data.identity"
```

Dan untuk menentukan node yang harus digunakan fungsi *`check-address`* Lambda, gunakan `InputPath` bidang sebagai berikut:

```
"InputPath": "$.data.address"
```

Sekarang jika Anda ingin menyimpan hasil verifikasi dalam input mesin negara asli, gunakan `ResultPath` bidang sebagai berikut:

```
"ResultPath": "$.results"
```

Namun, jika Anda hanya memerlukan identitas dan hasil verifikasi dan membuang input asli, gunakan `OutputPath` kolom sebagai berikut:

```
"OutputPath": "$.results"
```

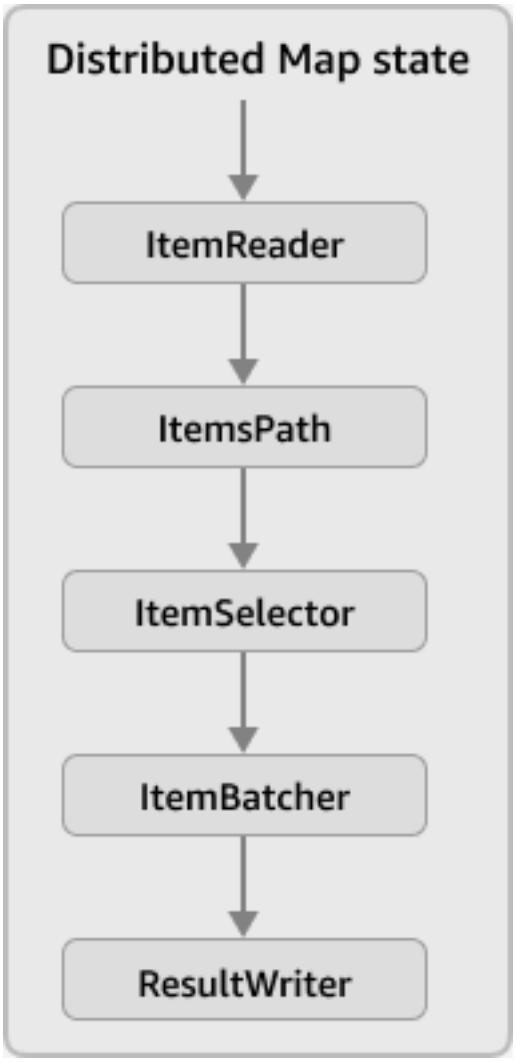
Untuk informasi selengkapnya, lihat Pengolahan Input dan Output di Step Functions.

Bidang input dan output negara peta

`Map`menyatakan secara bersamaan iterasi atas koleksi item dalam kumpulan data, seperti array JSON, daftar objek Amazon S3, atau baris file CSV dalam bucket Amazon S3. Ini mengulangi serangkaian langkah untuk setiap item dalam koleksi. Anda dapat mengkonfigurasi input yang diterima `Map` negara dan output yang dihasilkannya menggunakan bidang ini. Langkah Fungsi berlaku setiap bidang di negara Peta Terdistribusi Anda dalam urutan yang ditunjukkan dalam daftar berikut dan ilustrasi:

> **Note**
>
> Berdasarkan kasus penggunaan Anda, Anda mungkin tidak perlu menerapkan semua bidang ini.

1. ItemReader
2. ItemsPath
3. ItemSelector
4. ItemBatcher
5. ResultWriter



> **Note**
>
> Bidang input dan output `Map` status ini saat ini tidak tersedia di simulator aliran data di konsol Step Functions.

ItemReader

The`ItemReader`field adalah objek JSON, yang menentukan dataset dan lokasinya. SEBUAHStatus Peta Terdistribusimenggunakan dataset ini sebagai inputnya. Contoh berikut menunjukkan sintaks dari`ItemReader`field jika kumpulan data Anda adalah file CSV yang disimpan di bucket Amazon S3.

```
"ItemReader": {
    "ReaderConfig": {
```

```
        "InputType": "CSV",
        "CSVHeaderLocation": "FIRST_ROW"
    },
    "Resource": "arn:aws:states:::s3:getObject",
    "Parameters": {
        "Bucket": "myBucket",
        "Key": "csvDataset/ratings.csv"
    }
}
```

> **Tip**
>
> Di Workflow Studio, Anda menentukan kumpulan data dan lokasinya diSumber baranglapangan.

Daftar Isi

- Isi ItemReader bidang
- Contoh dataset
- Kebijakan IAM untuk kumpulan data

Isi ItemReader bidang

Tergantung pada dataset Anda, isi`ItemReader`bidang bervariasi. Misalnya, jika kumpulan data Anda adalah array JSON yang diteruskan dari langkah sebelumnya dalam alur kerja,`ItemReader`bidang dihilangkan. Jika kumpulan data Anda adalah sumber data Amazon S3, bidang ini berisi sub-bidang berikut.

`ReaderConfig`

Sebuah objek JSON yang menentukan rincian berikut:

- `InputType`

  Menentukan jenis sumber data Amazon S3, seperti file CSV, objek, file JSON, atau daftar inventaris Amazon S3. Di Workflow Studio, Anda dapat memilih jenis input dariSumber item Amazon S3daftar dropdown di bawahSumber baranglapangan.

- `CSVHeaderLocation`

> **Note**
>
> Anda harus menentukan bidang ini hanya jika Anda menggunakan file CSV sebagai kumpulan data.

Menerima salah satu nilai berikut untuk menentukan lokasi header kolom:

> **Important**
>
> Saat ini, Step Functions mendukung header CSV hingga 10 KB.

- `FIRST_ROW`— Gunakan opsi ini jika baris pertama file adalah header.
- `GIVEN`— Gunakan opsi ini untuk mengatur header di definisi mesin negara. Misalnya, jika file CSV Anda berisi data berikut.

```
1,307,3.5,1256677221
1,481,3.5,1256677456
1,1091,1.5,1256677471
...
```

Berikan array JSON berikut sebagai header CSV.

```
"ItemReader": {
    "ReaderConfig": {
        "InputType": "CSV",
        "CSVHeaderLocation": "GIVEN",
        "CSVHeaders": [
            "userId",
            "movieId",
            "rating",
            "timestamp"
        ]
    }
}
```

> **Tip**
>
> Di Workflow Studio, Anda dapat menemukan opsi ini di bawahKonfigurasi tambahandiSumber baranglapangan.

- `MaxItems`

  Membatasi jumlah item data yang dilewatkan ke`Map`negara. Misalnya, Anda menyediakan file CSV yang berisi 1000 baris dan tentukan batas 100. Kemudian, penerjemah lewatcuma100 baris ke`Map`negara. The`Map`status memproses item dalam urutan berurutan, dimulai setelah baris header.

  Secara default,`Map`state iterasi atas semua item dalam dataset yang ditentukan.

  > **Note**
  >
  > Saat ini, Anda dapat menentukan batas hingga 100.000.000. TheStatus Peta Terdistribusiberhenti membaca item di luar batas ini.

  > **Tip**
  >
  > Di Workflow Studio, Anda dapat menemukan opsi ini di bawahKonfigurasi tambahandiSumber baranglapangan.

  Atau, Anda dapat menentukanjalur referensike pasangan kunci-nilai yang ada di AndaStatus Peta Terdistribusimasukan. Jalur ini harus menyelesaikan ke bilangan bulat positif. Anda menentukan jalur referensi di`MaxItemsPath`sub-bidang.

  > **Important**
  >
  > Anda dapat menentukan salah satu`MaxItems`atau`MaxItemsPath`sub-bidang, tetapi tidak keduanya.

`Resource`

Tindakan Amazon S3 API yang harus dijalankan oleh Step Functions tergantung pada kumpulan data yang ditentukan.

`Parameters`

Objek JSON yang menentukan nama bucket Amazon S3 dan kunci objek tempat kumpulan data disimpan.

> **Important**
>
> Pastikan bucket Amazon S3 Anda berada di bawah yang samaAkun AWSdanWilayah AWSsebagai mesin negara.

Contoh dataset

Anda dapat mengatur salah satu opsi berikut sebagai kumpulan data Anda:

- JSON array dari langkah sebelumnya
- Daftar objek Amazon S3
- File JSON dalam ember Amazon S3
- File CSV dalam ember Amazon S3
- Daftar inventaris Amazon S3

> **Important**
>
> Step Functions memerlukan izin yang sesuai untuk mengakses kumpulan data Amazon S3 yang Anda gunakan. Untuk informasi tentang kebijakan IAM untuk kumpulan data, lihatKebijakan IAM untuk kumpulan data.

JSON array dari langkah sebelumnya

SEBUAHStatus Peta Terdistribusidapat menerima input JSON yang dilewatkan dari langkah sebelumnya di alur kerja. Masukan ini harus berupa array, atau harus berisi array dalam node tertentu. Untuk memilih node yang berisi array, Anda dapat menggunakan`ItemsPath`lapangan.

Untuk memproses item individual dalam array,Status Peta Terdistribusimemulai eksekusi alur kerja anak untuk setiap item array. Tab berikut menunjukkan contoh input yang diteruskan ke`Map`state dan input yang sesuai untuk eksekusi alur kerja anak.

> **Note**
>
> Step Functions menghilangkan`ItemReader`bidang saat dataset Anda adalah array JSON dari langkah sebelumnya.

Input passed to the Map state

Pertimbangkan array JSON berikut dari tiga item.

```
"facts": [
    {
        "verdict": "true",
        "statement_date": "6/11/2008",
        "statement_source": "speech"
    },
    {
        "verdict": "false",
        "statement_date": "6/7/2022",
        "statement_source": "television"
    },
    {
        "verdict": "mostly-true",
        "statement_date": "5/18/2016",
        "statement_source": "news"
    }
]
```

Input passed to a child workflow execution

TheStatus Peta Terdistribusimemulai eksekusi tiga alur kerja anak. Setiap eksekusi menerima item array sebagai input. Contoh berikut menunjukkan input yang diterima oleh eksekusi alur kerja anak.

```
{
  "verdict": "true",
  "statement_date": "6/11/2008",
```

```
  "statement_source": "speech"
}
```

Contoh objek Amazon S3

SEBUAHStatus Peta Terdistribusidapat mengulang objek yang disimpan di bucket Amazon S3. Ketika eksekusi alur kerja mencapai`Map`status, Step Functions memanggilListObjectsV2Tindakan API, yang mengembalikan array metadata objek Amazon S3. Dalam array ini, setiap item berisi data, sepertiETagdanKunci, untuk data yang disimpan di bucket.

Untuk memproses item individual dalam array,Status Peta Terdistribusimemulai eksekusi alur kerja anak. Misalnya, bucket Amazon S3 Anda berisi 100 gambar. Kemudian, array kembali setelah memanggil`ListObjectsV2`Tindakan API berisi 100 item. TheStatus Peta Terdistribusikemudian mulai 100 eksekusi alur kerja anak untuk memproses setiap item array.

> **Note**
>
> - Saat ini, Step Functions juga menyertakan item untuk setiap folder yang Anda buat di bucket Amazon S3 menggunakan konsol Amazon S3. Ini menghasilkan eksekusi alur kerja anak tambahan yang dimulai olehStatus Peta Terdistribusi. Untuk menghindari membuat eksekusi alur kerja anak tambahan untuk folder, kami sarankan Anda menggunakanAWS CLIuntuk membuat folder. Untuk informasi selengkapnya, lihatPerintah Amazon S3 tingkat tinggidiAWS Command Line InterfacePanduan Pengguna.
> - Step Functions memerlukan izin yang sesuai untuk mengakses kumpulan data Amazon S3 yang Anda gunakan. Untuk informasi tentang kebijakan IAM untuk kumpulan data, lihatKebijakan IAM untuk kumpulan data.

Tab berikut menunjukkan contoh`ItemReader`sintaks bidang dan input diteruskan ke eksekusi alur kerja anak untuk kumpulan data ini.

ItemReader syntax

Dalam contoh ini, Anda telah mengatur data Anda, yang mencakup gambar, file JSON, dan objek, dalam awalan bernama`processData`adalam ember Amazon S3 bernama`myBucket`.

```
"ItemReader": {
```

```
    "Resource": "arn:aws:states:::s3:listObjectsV2",
    "Parameters": {
        "Bucket": "myBucket",
        "Prefix": "processData"
    }
}
```

Input passed to a child workflow execution

TheStatus Peta Terdistribusimemulai eksekusi alur kerja anak sebanyak jumlah item di bucket Amazon S3. Contoh berikut menunjukkan input yang diterima oleh eksekusi alur kerja anak.

```
{
  "Etag": "\"05704fbdccb224cb01c59005bebbad28\"",
  "Key": "processData/images/n02085620_1073.jpg",
  "LastModified": 1668699881,
  "Size": 34910,
  "StorageClass": "STANDARD"
}
```

File JSON dalam ember Amazon S3

SEBUAHStatus Peta Terdistribusidapat menerima file JSON yang disimpan di bucket Amazon S3 sebagai kumpulan data. File JSON harus berisi array.

Ketika eksekusi alur kerja mencapai`Map`status, Step Functions memanggilGetObjectTindakan API untuk mengambil file JSON yang ditentukan. The`Map`state kemudian iterasi atas setiap item dalam array dan memulai eksekusi alur kerja anak untuk setiap item. Misalnya, jika file JSON Anda berisi 1000 item array,`Map`status memulai eksekusi alur kerja 1000 anak.

> **Note**
>
> - Input eksekusi yang digunakan untuk memulai eksekusi alur kerja anak tidak dapat melebihi 256 KB. Namun, Step Functions mendukung membaca item hingga 8 MB dari file CSV atau JSON jika Anda kemudian menerapkan opsional`ItemSelector`bidang untuk mengurangi ukuran item.
> - Saat ini, Step Functions mendukung 10 GB sebagai ukuran maksimum file individual dalam laporan inventaris Amazon S3. Namun, Step Functions dapat memproses lebih dari 10 GB jika setiap file individu di bawah 10 GB.

- Step Functions memerlukan izin yang sesuai untuk mengakses kumpulan data Amazon S3 yang Anda gunakan. Untuk informasi tentang kebijakan IAM untuk kumpulan data, lihatKebijakan IAM untuk kumpulan data.

Tab berikut menunjukkan contoh`ItemReader`sintaks bidang dan input diteruskan ke eksekusi alur kerja anak untuk kumpulan data ini.

Untuk contoh ini, bayangkan Anda memiliki file JSON bernama*`factcheck.json`*. Anda telah menyimpan file ini dalam awalan bernama*`jsonDataset`*di ember Amazon S3. Berikut ini adalah contoh dari dataset JSON.

```
[
  {
    "verdict": "true",
    "statement_date": "6/11/2008",
    "statement_source": "speech"
  },
  {
    "verdict": "false",
    "statement_date": "6/7/2022",
    "statement_source": "television"
  },
  {
    "verdict": "mostly-true",
    "statement_date": "5/18/2016",
    "statement_source": "news"
  },
  ...
]
```

ItemReader syntax

```
"ItemReader": {
    "Resource": "arn:aws:states:::s3:getObject",
    "ReaderConfig": {
        "InputType": "JSON"
    },
    "Parameters": {
        "Bucket": "myBucket",
        "Key": "jsonDataset/factcheck.json"
```

```
    }
}
```

Input to a child workflow execution

TheStatus Peta Terdistribusimemulai eksekusi alur kerja anak sebanyak jumlah item array yang ada dalam file JSON. Contoh berikut menunjukkan input yang diterima oleh eksekusi alur kerja anak.

```
{
  "verdict": "true",
  "statement_date": "6/11/2008",
  "statement_source": "speech"
}
```

File CSV dalam ember Amazon S3

SEBUAHStatus Peta Terdistribusidapat menerima file CSV yang disimpan di bucket Amazon S3 sebagai kumpulan data. Jika Anda menggunakan file CSV sebagai kumpulan data Anda, Anda perlu menentukan header kolom CSV. Untuk informasi tentang cara mengatur header CSV, lihatIsi ItemReader bidang.

Karena tidak ada format standar untuk membuat dan memelihara data dalam file CSV, Step Functions mem-parsing file CSV berdasarkan aturan berikut:

- Koma (,) adalah pembatas yang memisahkan bidang individu.
- Baris baru adalah pembatas yang memisahkan catatan individu.
- Bidang diperlakukan sebagai string. Untuk konversi tipe data, gunakan`States.StringToJson`fungsi intrinsikItemSelector.
- Tanda kutip ganda (” “) tidak diperlukan untuk menyertakan string. Namun, string yang dilampirkan oleh tanda kutip ganda dapat berisi koma dan baris baru tanpa berfungsi sebagai pembatas.
- Melarikan diri dari tanda kutip ganda dengan mengulanginya.
- Jika jumlah bidang dalam satu baris kurang dari jumlah bidang di header, Step Functions menyediakan string kosong untuk nilai yang hilang.
- Jika jumlah bidang dalam satu baris lebih dari jumlah bidang di header, Step Functions melewatkan bidang tambahan.

Untuk informasi selengkapnya tentang cara Step Functions mengurai file CSV, lihatExample of parsing an input CSV file.

Ketika eksekusi alur kerja mencapaiMapstatus, Step Functions memanggilGetObjectTindakan API untuk mengambil file CSV yang ditentukan. TheMapstate kemudian iterasi di setiap baris dalam file CSV dan memulai eksekusi alur kerja anak untuk memproses item di setiap baris. Misalnya, Anda menyediakan file CSV yang berisi 100 baris sebagai input. Kemudian, penerjemah meneruskan setiap baris keMapnegara. TheMapstatus memproses item dalam urutan serial, dimulai setelah baris header.

> **Note**
>
> - Input eksekusi yang digunakan untuk memulai eksekusi alur kerja anak tidak dapat melebihi 256 KB. Namun, Step Functions mendukung membaca item hingga 8 MB dari file CSV atau JSON jika Anda kemudian menerapkan opsionalItemSelectorbidang untuk mengurangi ukuran item.
> - Saat ini, Step Functions mendukung 10 GB sebagai ukuran maksimum file individual dalam laporan inventaris Amazon S3. Namun, Step Functions dapat memproses lebih dari 10 GB jika setiap file individu di bawah 10 GB.
> - Step Functions memerlukan izin yang sesuai untuk mengakses kumpulan data Amazon S3 yang Anda gunakan. Untuk informasi tentang kebijakan IAM untuk kumpulan data, lihatKebijakan IAM untuk kumpulan data.

Tab berikut menunjukkan contohItemReadersintaks bidang dan input diteruskan ke eksekusi alur kerja anak untuk kumpulan data ini.

ItemReader syntax

Misalnya, Anda memiliki file CSV bernama*ratings.csv*. Kemudian, Anda telah menyimpan file ini dalam awalan yang diberi nama*csvDataset*di ember Amazon S3.

```
{
  "ItemReader": {
    "ReaderConfig": {
      "InputType": "CSV",
      "CSVHeaderLocation": "FIRST_ROW"
    },
    "Resource": "arn:aws:states:::s3:getObject",
```

```
    "Parameters": {
      "Bucket": "myBucket",
      "Key": "csvDataset/ratings.csv"
    }
  }
}
```

Input to a child workflow execution

TheStatus Peta Terdistribusimemulai eksekusi alur kerja anak sebanyak jumlah baris yang ada dalam file CSV, tidak termasuk baris header, jika dalam file. Contoh berikut menunjukkan input yang diterima oleh eksekusi alur kerja anak.

```
{
  "rating": "3.5",
  "movieId": "307",
  "userId": "1",
  "timestamp": "1256677221"
}
```

Contoh inventaris S3

SEBUAHStatus Peta Terdistribusidapat menerima file manifes inventaris Amazon S3 yang disimpan di bucket Amazon S3 sebagai kumpulan data.

Ketika eksekusi alur kerja mencapaiMapstatus, Step Functions memanggilGetObjectTindakan API untuk mengambil file manifes inventaris Amazon S3 yang ditentukan. TheMapstate kemudian mengulangi objek dalam inventaris untuk mengembalikan array metadata objek inventaris Amazon S3.

> **Note**
>
> - Saat ini, Step Functions mendukung 10 GB sebagai ukuran maksimum file individual dalam laporan inventaris Amazon S3. Namun, Step Functions dapat memproses lebih dari 10 GB jika setiap file individu di bawah 10 GB.
> - Step Functions memerlukan izin yang sesuai untuk mengakses kumpulan data Amazon S3 yang Anda gunakan. Untuk informasi tentang kebijakan IAM untuk kumpulan data, lihatKebijakan IAM untuk kumpulan data.

Berikut ini adalah contoh file inventaris di format CSV. File ini mencakup objek bernama`csvDataset`dan`imageDataset`, yang disimpan di bucket Amazon S3 yang diberi nama`sourceBucket`.

```
"sourceBucket","csvDataset/","0","2022-11-16T00:27:19.000Z"
"sourceBucket","csvDataset/titles.csv","3399671","2022-11-16T00:29:32.000Z"
"sourceBucket","imageDataset/","0","2022-11-15T20:00:44.000Z"
"sourceBucket","imageDataset/n02085620_10074.jpg","27034","2022-11-15T20:02:16.000Z"
...
```

> **Important**
>
> Saat ini, Step Functions tidak mendukung laporan inventaris Amazon S3 yang ditentukan pengguna sebagai kumpulan data. Anda juga harus memastikan bahwa format keluaran laporan inventaris Amazon S3 Anda adalah CSV. Untuk informasi selengkapnya tentang inventaris Amazon S3 dan cara mengatur inventaris Amazon S3, lihatInventaris Amazon S3diPanduan Pengguna Amazon S3.

Contoh berikut dari file manifes inventaris menunjukkan header CSV untuk metadata objek inventaris.

```
{
  "sourceBucket" : "sourceBucket",
  "destinationBucket" : "arn:aws:s3:::inventory",
  "version" : "2016-11-30",
  "creationTimestamp" : "1668560400000",
  "fileFormat" : "CSV",
  "fileSchema" : "Bucket, Key, Size, LastModifiedDate",
  "files" : [ {
    "key" : "source-bucket/destination-prefix/
data/20e55de8-9c21-45d4-99b9-46c732000228.csv.gz",
    "size" : 7300,
    "MD5checksum" : "a7ff4a1d4164c3cd55851055ec8f6b20"
  } ]
}
```

Tab berikut menunjukkan contoh`ItemReader`sintaks bidang dan input diteruskan ke eksekusi alur kerja anak untuk kumpulan data ini.

ItemReader syntax

```
{
  "ItemReader": {
    "ReaderConfig": {
      "InputType": "MANIFEST"
    },
    "Resource": "arn:aws:states:::s3:getObject",
    "Parameters": {
      "Bucket": "destinationBucket",
      "Key": "destination-prefix/source-bucket/config-ID/YYYY-MM-DDTHH-MMZ/
manifest.json"
    }
  }
}
```

Input to a child workflow execution

```
{
  "LastModifiedDate": "2022-11-16T00:29:32.000Z",
  "Bucket": "sourceBucket",
  "Size": "3399671",
  "Key": "csvDataset/titles.csv"
}
```

Bergantung pada bidang yang Anda pilih saat mengonfigurasi laporan inventaris Amazon S3, kontenmanifest.jsonfile dapat bervariasi dari contoh yang ditampilkan.

Kebijakan IAM untuk kumpulan data

Saat Anda membuat alur kerja dengan konsol Step Functions, Step Functions dapat secara otomatis menghasilkan kebijakan IAM berdasarkan sumber daya dalam definisi alur kerja Anda. Kebijakan ini mencakup hak istimewa paling sedikit yang diperlukan untuk memungkinkan peran mesin negara untuk memanggilStartExecutionTindakan API untukStatus Peta Terdistribusi. Kebijakan ini juga mencakup keistimewaan paling sedikit Step Functions yang diperlukan untuk diaksesAWSsumber daya, seperti bucket Amazon S3 dan fungsi Lambda. Kami sangat menyarankan Anda untuk menyertakan izin yang diperlukan di kebijakan IAM Anda. Misalnya, jika alur kerja Anda menyertakanMapstatus dalam mode Terdistribusi, cakup kebijakan Anda ke bucket Amazon S3 tertentu dan folder yang berisi kumpulan data Anda.

> **Important**
>
> Jika Anda menentukan bucket dan objek Amazon S3, atau awalan, denganjalur referensike pasangan kunci-nilai yang ada di AndaStatus Peta Terdistribusimasukan, pastikan Anda memperbarui kebijakan IAM untuk alur kerja Anda. Cakupan kebijakan hingga ke bucket dan nama objek yang diselesaikan jalur saat runtime.

Contoh kebijakan IAM berikut memberikan hak istimewa paling sedikit yang diperlukan untuk mengakses kumpulan data Amazon S3 Anda menggunakanListObjectsV2danGetObjectTindakan API.

Example Kebijakan IAM untuk objek Amazon S3 sebagai kumpulan data

Contoh berikut menunjukkan kebijakan IAM yang memberikan izin paling sedikit untuk mengakses objek yang diatur di dalamnya*processImages*dalam ember Amazon S3 bernama*myBucket*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket"
            ],
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
                        "processImages"
                    ]
                }
            }
        }
    ]
}
```

Example Kebijakan IAM untuk file CSV sebagai kumpulan data

Contoh berikut menunjukkan kebijakan IAM yang memberikan izin paling sedikit untuk mengakses file CSV bernama*ratings.csv*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket/csvDataset/ratings.csv"
            ]
        }
    ]
}
```

Example Kebijakan IAM untuk inventaris Amazon S3 sebagai kumpulan data

Contoh berikut menunjukkan kebijakan IAM yang memberikan izin paling sedikit untuk mengakses laporan inventaris Amazon S3.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::destination-prefix/source-bucket/config-ID/YYYY-MM-DDTHH-
MMZ/manifest.json",
                "arn:aws:s3:::destination-prefix/source-bucket/config-ID/data/*"
            ]
        }
    ]
}
```

ItemsPath

Gunakan `ItemsPath` bidang untuk memilih array dalam masukan JSON yang disediakan untuk `Map` negara. `Map`Negara mengulangi serangkaian langkah untuk setiap item dalam array. Secara default, `Map` negara set `ItemsPath` ke`$`, yang memilih seluruh masukan. Jika input ke `Map` state adalah array JSON, ia menjalankan iterasi untuk setiap item dalam array, meneruskan item itu ke iterasi sebagai input.

> **Note**
>
> Anda dapat menggunakan `ItemsPath` status Peta Terdistribusi hanya jika Anda menggunakan masukan JSON yang diteruskan dari keadaan sebelumnya dalam alur kerja.

Anda dapat menggunakan `ItemsPath` bidang untuk menentukan lokasi di input yang menunjuk ke array JSON yang digunakan untuk iterasi. Nilai `ItemsPath` harus berupa Reference Path, dan path yang harus menunjuk ke array JSON. Misalnya, pertimbangkan input ke status `Map` yang mencakup dua array, seperti contoh berikut.

```
{
  "ThingsPiratesSay": [
    {
      "say": "Avast!"
    },
    {
      "say": "Yar!"
    },
    {
      "say": "Walk the Plank!"
    }
  ],
  "ThingsGiantsSay": [
    {
      "say": "Fee!"
    },
    {
      "say": "Fi!"
    },
    {
      "say": "Fo!"
    },
```

```
    {
      "say": "Fum!"
    }
  ]
}
```

Dalam kasus ini, Anda dapat menentukan array mana yang akan digunakan untuk iterasi `Map` negara dengan `ItemsPath` memilihnya. Definisi mesin negara berikut menentukan `ThingsPiratesSay` array dalam input `ItemsPath` menggunakan.Ini kemudian menjalankan iterasi dari negara `SayWord` lulus untuk setiap item dalam array. `ThingsPiratesSay`

```
{
  "StartAt": "PiratesSay",
  "States": {
    "PiratesSay": {
      "Type": "Map",
      "ItemsPath": "$.ThingsPiratesSay",
      "ItemProcessor": {
         "StartAt": "SayWord",
         "States": {
           "SayWord": {
             "Type": "Pass",
             "End": true
           }
         }
      },
      "End": true
    }
  }
}
```

Saat memproses input, `Map` negara berlaku `ItemsPath` setelahnya `InputPath`. Ini beroperasi pada input yang efektif untuk negara setelah `InputPath` filter input.

Untuk informasi selengkapnya tentang status `Map`, lihat hal berikut:

- Negara peta
- Mode pemrosesan status peta
- Ulangi tindakan menggunakan status Peta Inline
- Pemrosesan input dan output `Map` status sebaris

ItemSelector

Secara default, input efektif untuk `Map` negara adalah kumpulan item data individual yang ada dalam masukan status mentah. `ItemSelector`Bidang ini memungkinkan Anda mengganti nilai item data sebelum diteruskan ke state. `Map` Untuk mengganti nilai, tentukan input JSON yang valid yang berisi kumpulan pasangan kunci-nilai. Pasangan ini dapat berupa nilai statis yang disediakan dalam definisi mesin negara Anda, nilai yang dipilih dari masukan negara menggunakan jalur, atau nilai yang diakses dari objek konteks.

Jika Anda menentukan pasangan kunci-nilai menggunakan path atau objek konteks, nama kunci harus diakhiri. `.$`

> **Note**
>
> `ItemSelector`Bidang menggantikan `Parameters` bidang dalam `Map` negara. Jika Anda menggunakan `Parameters` bidang dalam definisi `Map` status Anda untuk membuat input kustom, kami sangat menyarankan Anda menggantinya dengan`ItemSelector`.

Anda dapat menentukan `ItemSelector` bidang dalam status Peta Inline dan status Peta Terdistribusi.

Misalnya, pertimbangkan input JSON berikut yang berisi array tiga item dalam `imageData` node. Untuk setiap iterasi *`Map`* negara, item array diteruskan ke iterasi sebagai masukan.

```
[
  {
    "resize": "true",
    "format": "jpg"
  },
  {
    "resize": "false",
    "format": "png"
  },
  {
    "resize": "true",
    "format": "jpg"
  }
]
```

Menggunakan `ItemSelector` bidang, Anda dapat menentukan masukan JSON kustom untuk menimpa masukan asli seperti yang ditunjukkan pada contoh berikut. Langkah Fungsi kemudian melewati masukan kustom ini untuk setiap iterasi *Map* negara. Masukan kustom berisi nilai statis untuk `size` dan nilai data objek konteks untuk `Map` negara. Objek `$$.Map.Item.Value` konteks berisi nilai setiap item data individu.

```
{
  "ItemSelector": {
    "size": 10,
    "value.$": "$$.Map.Item.Value"
  }
}
```

Contoh berikut menunjukkan masukan yang diterima oleh satu iterasi dari negara Inline Peta:

```
{
  "size": 10,
  "value": {
    "resize": "true",
    "format": "jpg"
  }
}
```

> **Tip**
>
> Untuk contoh lengkap status Peta Terdistribusi yang menggunakan `ItemSelector` bidang, lihatMemulai dengan menggunakan status Peta Terdistribusi.

ItemBatcher

`ItemBatcher`Bidang adalah objek JSON, yang menentukan untuk memproses sekelompok item dalam eksekusi alur kerja anak tunggal. Gunakan pengelompokan saat memproses file CSV besar atau array JSON, atau kumpulan besar objek Amazon S3.

Contoh berikut menunjukkan sintaks `ItemBatcher` lapangan. Dalam sintaks berikut, jumlah maksimum item yang harus diproses oleh setiap eksekusi alur kerja anak diatur ke 100.

```
{
  "ItemBatcher": {
```

```
    "MaxItemsPerBatch": 100
  }
}
```

Secara default, setiap item dalam dataset dilewatkan sebagai masukan ke eksekusi alur kerja anak individu. Misalnya, asumsikan Anda menentukan file JSON sebagai masukan yang berisi array berikut:

```
[
  {
    "verdict": "true",
    "statement_date": "6/11/2008",
    "statement_source": "speech"
  },
  {
    "verdict": "false",
    "statement_date": "6/7/2022",
    "statement_source": "television"
  },
  {
    "verdict": "true",
    "statement_date": "5/18/2016",
    "statement_source": "news"
  },
  ...
]
```

Untuk masukan yang diberikan, setiap eksekusi alur kerja anak menerima item array sebagai masukan. Contoh berikut menunjukkan masukan dari eksekusi alur kerja anak:

```
{
  "verdict": "true",
  "statement_date": "6/11/2008",
  "statement_source": "speech"
}
```

Untuk membantu mengoptimalkan kinerja dan biaya pekerjaan pemrosesan Anda, pilih ukuran batch yang menyeimbangkan jumlah item terhadap waktu pemrosesan item. Jika Anda menggunakan batching, Step Functions menambahkan item ke larik Item. Kemudian melewati array sebagai masukan untuk setiap eksekusi alur kerja anak. Contoh berikut menunjukkan batch dua item lulus sebagai masukan untuk eksekusi alur kerja anak:

```
{
  "Items": [
    {
      "verdict": "true",
      "statement_date": "6/11/2008",
      "statement_source": "speech"
    },
    {
      "verdict": "false",
      "statement_date": "6/7/2022",
      "statement_source": "television"
    }
  ]
}
```

Tip

Untuk mempelajari lebih lanjut tentang menggunakan `ItemBatcher` bidang dalam alur kerja Anda, coba tutorial dan lokakarya berikut:

- Memproses seluruh kumpulan data dalam fungsi Lambda
- Iterasi atas item dalam batch di dalam eksekusi alur kerja anak
- Paralelisasi Skala Besar dengan Peta Terdistribusi pada Modul 14 - Pemrosesan Data Lokakarya AWS Step Functions

Daftar Isi

- Bidang untuk menentukan pengelompokan item

Bidang untuk menentukan pengelompokan item

Untuk batch item, tentukan jumlah maksimum item untuk batch, ukuran batch maksimum, atau keduanya. Anda harus menentukan salah satu nilai ini ke item batch.

Item maks per batch

Menentukan jumlah maksimum item yang setiap proses eksekusi alur kerja anak. Interpreter membatasi jumlah item yang dikelompokkan dalam `Items` array ke nilai ini. Jika Anda

menentukan nomor dan ukuran batch, penerjemah mengurangi jumlah item dalam batch untuk menghindari melebihi batas ukuran batch yang ditentukan.

Jika Anda tidak menentukan nilai ini tetapi memberikan nilai untuk ukuran batch maksimum, Step Functions memproses item sebanyak mungkin dalam setiap eksekusi alur kerja anak tanpa melebihi ukuran batch maksimum dalam byte.

Misalnya, bayangkan Anda menjalankan eksekusi dengan file JSON masukan yang berisi 1130 node. Jika Anda menentukan nilai item maksimum untuk setiap batch 100, Step Functions membuat 12 batch. Dari jumlah tersebut, 11 batch berisi 100 item masing-masing, sedangkan batch kedua belas berisi 30 item yang tersisa.

Atau, Anda dapat menentukan item maksimum untuk setiap batch sebagai jalur referensi ke pasangan nilai kunci yang ada di masukan status Peta Terdistribusi Anda. Jalur ini harus menyelesaikan ke bilangan bulat positif.

Misalnya, diberikan input berikut:

```
{
  "maxBatchItems": 500
}
```

Anda dapat menentukan jumlah maksimum item yang akan dikumpulkan menggunakan jalur referensi sebagai berikut:

```
{
  ...
  "Map": {
    "Type": "Map",
    "MaxConcurrency": 2000,
    "ItemBatcher": {
      "MaxItemsPerBatchPath": "$.maxBatchItems"
    }
    ...
    ...
  }
}
```

> **Important**
>
> Anda dapat menentukan `MaxItemsPerBatch` atau `MaxItemsPerBatchPath` sub-bidang, tetapi tidak keduanya.

Max KBs per batch

Menentukan ukuran maksimum batch dalam byte, hingga 256 KBs. Jika Anda menentukan jumlah dan ukuran batch maksimum, Step Functions mengurangi jumlah item dalam batch agar tidak melebihi batas ukuran batch yang ditentukan.

Atau, Anda dapat menentukan ukuran batch maksimum sebagai jalur referensi ke pasangan kunci-nilai yang ada di masukan status Peta Terdistribusi Anda. Jalur ini harus menyelesaikan ke bilangan bulat positif.

> **Note**
>
> Jika Anda menggunakan batching dan tidak menentukan ukuran batch maksimum, interpreter memproses sebanyak mungkin item yang dapat diproses hingga 256 KB di setiap eksekusi alur kerja anak.

Misalnya, diberikan input berikut:

```
{
  "batchSize": 131072
}
```

Anda dapat menentukan ukuran batch maksimum menggunakan jalur referensi sebagai berikut:

```
{
  ...
  "Map": {
    "Type": "Map",
    "MaxConcurrency": 2000,
    "ItemBatcher": {
      "MaxInputBytesPerBatchPath": "$.batchSize"
    }
    ...
```

```
    ...
  }
}
```

> **Important**
>
> Anda dapat menentukan `MaxInputBytesPerBatch` atau `MaxInputBytesPerBatchPath` sub-bidang, tetapi tidak keduanya.

Masukan batch

Secara opsional, Anda juga dapat menentukan input JSON tetap untuk disertakan dalam setiap batch yang diteruskan ke setiap eksekusi alur kerja anak. Langkah Fungsi menggabungkan masukan ini dengan masukan untuk setiap eksekusi alur kerja anak individu. Misalnya, mengingat masukan tetap berikut dari tanggal pemeriksaan fakta pada array item:

```
"ItemBatcher": {
    "BatchInput": {
        "factCheck": "December 2022"
    }
}
```

Setiap eksekusi alur kerja anak menerima berikut sebagai masukan:

```
{
  "BatchInput": {
    "factCheck": "December 2022"
  },
  "Items": [
    {
      "verdict": "true",
      "statement_date": "6/11/2008",
      "statement_source": "speech"
    },
    {
      "verdict": "false",
      "statement_date": "6/7/2022",
      "statement_source": "television"
    },
    ...
```

```
  ]
}
```

ResultWriter

`ResultWriter`Bidang adalah objek JSON yang menentukan lokasi Amazon S3 tempat Step Functions menulis hasil eksekusi alur kerja anak yang dimulai oleh status Peta Terdistribusi. Secara default, Step Functions tidak mengekspor hasil ini.

> **Important**
>
> Pastikan bucket Amazon S3 yang Anda gunakan untuk mengekspor hasil Map Run berada di bawah yang sama Akun AWS dan Wilayah AWS sebagai mesin status Anda. Jika tidak, eksekusi mesin status Anda akan gagal dengan `States.ResultWriterFailed` kesalahan tersebut.

Mengekspor hasil ke bucket Amazon S3 sangat membantu jika ukuran payload output Anda melebihi 256 KB. Step Functions menggabungkan semua data eksekusi alur kerja anak, seperti input dan output eksekusi, ARN, dan status eksekusi. Kemudian mengekspor eksekusi dengan status yang sama ke file masing-masing di lokasi Amazon S3 yang ditentukan. Contoh berikut menunjukkan sintaks `ResultWriter` bidang jika Anda mengekspor hasil eksekusi alur kerja anak. Dalam contoh ini, Anda menyimpan hasilnya dalam bucket bernama `myOutputBucket` dalam awalan yang disebut`csvProcessJobs`.

```
{
  "ResultWriter": {
    "Resource": "arn:aws:states:::s3:putObject",
    "Parameters": {
      "Bucket": "myOutputBucket",
      "Prefix": "csvProcessJobs"
    }
  }
}
```

> **Tip**
>
> Di Workflow Studio, Anda dapat mengekspor hasil eksekusi alur kerja turunan dengan memilih Hasil status Peta Ekspor ke Amazon S3. Kemudian, berikan nama bucket Amazon S3 dan awalan tempat Anda ingin mengekspor hasilnya.

Step Functions membutuhkan izin yang sesuai untuk mengakses bucket dan folder tempat Anda ingin mengekspor hasilnya. Untuk informasi tentang kebijakan IAM yang diperlukan, lihatKebijakan IAM untuk ResultWriter.

Jika Anda mengekspor hasil eksekusi alur kerja turunan, eksekusi status Peta Terdistribusi akan menampilkan ARN Jalankan Peta dan data tentang lokasi ekspor Amazon S3 dalam format berikut:

```
{
  "MapRunArn": "arn:aws:states:us-
east-2:123456789012:mapRun:csvProcess/Map:ad9b5f27-090b-3ac6-9beb-243cd77144a7",
  "ResultWriterDetails": {
    "Bucket": "myOutputBucket",
    "Key": "csvProcessJobs/ad9b5f27-090b-3ac6-9beb-243cd77144a7/manifest.json"
  }
}
```

Step Functions mengekspor eksekusi dengan status yang sama ke file masing-masing. Misalnya, jika eksekusi alur kerja anak Anda menghasilkan 500 keberhasilan dan 200 hasil kegagalan, Step Functions akan membuat dua file di lokasi Amazon S3 yang ditentukan untuk hasil keberhasilan dan kegagalan. Dalam contoh ini, file hasil sukses berisi 500 hasil keberhasilan, sedangkan file hasil kegagalan berisi 200 hasil kegagalan.

Untuk upaya eksekusi tertentu, Step Functions membuat file berikut di lokasi Amazon S3 yang ditentukan tergantung pada output eksekusi Anda:

- `manifest.json`— Berisi metadata Map Run, seperti lokasi ekspor, Map Run ARN, dan informasi tentang file hasil.

  Jika Anda redrivenmemiliki Map Run, `manifest.json` file tersebut, berisi referensi ke semua eksekusi alur kerja anak yang berhasil di semua upaya Map Run. Namun, file ini berisi referensi ke eksekusi yang gagal dan tertunda untuk yang spesifikredrive.

- `SUCCEEDED_n.json`— Berisi data konsolidasi untuk semua eksekusi alur kerja anak yang berhasil. n mewakili nomor indeks file. Nomor indeks dimulai dari 0. Sebagai contoh, `SUCCEEDED_1.json`.
- `FAILED_n.json`— Berisi data konsolidasi untuk semua eksekusi alur kerja anak yang gagal, habis waktu, dan dibatalkan. Gunakan file ini untuk memulihkan dari eksekusi yang gagal. n mewakili indeks file. Nomor indeks dimulai dari 0. Sebagai contoh, `FAILED_1.json`.
- `PENDING_n.json`— Berisi data konsolidasi untuk semua eksekusi alur kerja anak yang tidak dimulai karena Map Run gagal atau dibatalkan. n mewakili indeks file. Nomor indeks dimulai dari 0. Sebagai contoh, `PENDING_1.json`.

Step Functions mendukung file hasil individual hingga 5 GB. Jika ukuran file melebihi 5 GB, Step Functions membuat file lain untuk menulis hasil eksekusi yang tersisa dan menambahkan nomor indeks ke nama file. Misalnya, jika ukuran `Succeeded_0.json` file melebihi 5 GB, Step Functions membuat `Succeeded_1.json` file untuk merekam hasil yang tersisa.

Jika Anda tidak menentukan untuk mengekspor hasil eksekusi alur kerja anak, eksekusi mesin status mengembalikan larik hasil eksekusi alur kerja anak seperti yang ditunjukkan pada contoh berikut:

> **Note**
>
> Jika ukuran output yang dikembalikan melebihi 256 KB, eksekusi mesin status gagal dan mengembalikan `States.DataLimitExceeded` kesalahan.

```
[
  {
    "statusCode": 200,
    "inputReceived": {
      "show_id": "s1",
      "release_year": "2020",
      "rating": "PG-13",
      "type": "Movie"
    }
  },
  {
    "statusCode": 200,
    "inputReceived": {
      "show_id": "s2",
      "release_year": "2021",
```

```
      "rating": "TV-MA",
      "type": "TV Show"
    }
  },
  ...
]
```

Kebijakan IAM untuk ResultWriter

Saat Anda membuat alur kerja dengan konsol Step Functions, Step Functions dapat secara otomatis menghasilkan kebijakan IAM berdasarkan sumber daya dalam definisi alur kerja Anda. Kebijakan ini mencakup hak istimewa paling sedikit yang diperlukan untuk memungkinkan peran mesin status menjalankan tindakan `StartExecution` API untuk status Peta Terdistribusi. Kebijakan ini juga mencakup hak istimewa paling sedikit Step Functions yang diperlukan untuk mengakses AWS sumber daya, seperti bucket dan objek Amazon S3 serta fungsi Lambda. Kami sangat menyarankan agar Anda hanya menyertakan izin yang diperlukan dalam kebijakan IAM Anda. Misalnya, jika alur kerja Anda menyertakan `Map` status dalam mode Terdistribusi, cakupkan kebijakan Anda ke bucket Amazon S3 tertentu dan folder yang berisi kumpulan data Anda.

> **Important**
>
> Jika Anda menentukan bucket dan objek Amazon S3, atau awalan, dengan jalur referensi ke pasangan nilai kunci yang ada di input status Peta Terdistribusi, pastikan Anda memperbarui kebijakan IAM untuk alur kerja Anda. Cakupan kebijakan hingga ke bucket dan nama objek yang diselesaikan jalur saat runtime.

Contoh kebijakan IAM berikut memberikan hak istimewa paling sedikit yang diperlukan untuk menulis hasil eksekusi alur kerja turunan Anda ke folder bernama *`CSVjobs di bucket`* Amazon S3 menggunakan tindakan API. `PutObject`

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListMultipartUploadParts",
```

```
                "s3:AbortMultipartUpload"
            ],
            "Resource": [
                "arn:aws:s3:::resultBucket/csvJobs/*"
            ]
        }
    ]
}
```

Jika bucket Amazon S3 tempat Anda menulis hasil eksekusi alur kerja turunan dienkripsi menggunakan AWS Key Management Service (AWS KMS) kunci, Anda harus menyertakan AWS KMS izin yang diperlukan dalam kebijakan IAM Anda. Untuk informasi selengkapnya, lihat Izin IAM untuk bucket AWS KMS key Amazon S3 terenkripsi.

Mengurai file CSV masukan

Karena tidak ada format standar untuk membuat dan memelihara data dalam file CSV, Step Functions mem-parsing file CSV berdasarkan aturan berikut:

- Koma (,) adalah pembatas yang memisahkan bidang individu.
- Baris baru adalah pembatas yang memisahkan catatan individu.
- Bidang diperlakukan sebagai string. Untuk konversi tipe data, gunakan`States.StringToJson`fungsi intrinsik diItemSelector.
- Tanda kutip ganda (“) tidak diperlukan untuk melampirkan string. Namun, string yang dilampirkan oleh tanda kutip ganda dapat berisi koma dan baris baru tanpa berfungsi sebagai pembatas.
- Melarikan diri dari tanda kutip ganda dengan mengulanginya.
- Jika jumlah bidang dalam satu baris kurang dari jumlah bidang di header, Step Functions menyediakan string kosong untuk nilai yang hilang.
- Jika jumlah bidang dalam satu baris lebih dari jumlah bidang di header, Step Functions melewatkan bidang tambahan.

Example penguraian file CSV masukan

Katakan bahwa Anda telah menyediakan file CSV bernama*myCSVInput.csv*yang berisi satu baris sebagai input. Kemudian, Anda telah menyimpan file di bucket Amazon S3 yang diberi nama*my-bucket*. File CSV adalah sebagai berikut.

```
abc,123,"This string contains commas, a double quotation marks (""), and a newline (
```

```
)",{""MyKey"":""MyValue""},"[1,2,3]"
```

Mesin status berikut membaca file CSV ini dan menggunakanItemSelectoruntuk mengonversi tipe data dari beberapa bidang.

```
{
  "StartAt": "Map",
  "States": {
    "Map": {
      "Type": "Map",
      "ItemProcessor": {
        "ProcessorConfig": {
          "Mode": "DISTRIBUTED",
          "ExecutionType": "STANDARD"
        },
        "StartAt": "Pass",
        "States": {
          "Pass": {
            "Type": "Pass",
            "End": true
          }
        }
      },
      "End": true,
      "Label": "Map",
      "MaxConcurrency": 1000,
      "ItemReader": {
        "Resource": "arn:aws:states:::s3:getObject",
        "ReaderConfig": {
          "InputType": "CSV",
          "CSVHeaderLocation": "GIVEN",
          "CSVHeaders": [
            "MyLetters",
            "MyNumbers",
            "MyString",
            "MyObject",
            "MyArray"
          ]
        },
        "Parameters": {
          "Bucket": "my-bucket",
          "Key": "myCSVInput.csv"
        }
```

```
      },
      "ItemSelector": {
        "MyLetters.$": "$$.Map.Item.Value.MyLetters",
        "MyNumbers.$": "States.StringToJson($$.Map.Item.Value.MyNumbers)",
        "MyString.$": "$$.Map.Item.Value.MyString",
        "MyObject.$": "States.StringToJson($$.Map.Item.Value.MyObject)",
        "MyArray.$": "States.StringToJson($$.Map.Item.Value.MyArray)"
      }
    }
  }
}
```

Ketika Anda menjalankan mesin status ini, menghasilkan output berikut.

```
[
  {
    "MyNumbers": 123,
    "MyObject": {
      "MyKey": "MyValue"
    },
    "MyString": "This string contains commas, a double quote (\"), and a newline (\n)",
    "MyLetters": "abc",
    "MyArray": [
      1,
      2,
      3
    ]
  }
]
```

Objek konteks

Objek konteks adalah struktur JSON internal yang tersedia selama eksekusi, dan berisi informasi tentang mesin dan eksekusi status Anda. Hal ini mengizinkan alur kerja Anda mengakses informasi tentang eksekusi spesifiknya. Anda dapat mengakses objek konteks dari bidang-bidang berikut:

- `InputPath`
- `OutputPath`
- `ItemsPath` (di status Peta)
- `Variable` (di status Pilihan)
- `ResultSelector`

- `Parameters`
- Operator perbandingan variabel ke variabel

Format Objek Konteks

Objek konteks mencakup informasi tentang mesin status, status, eksekusi, dan tugas. Objek JSON ini mencakup simpul untuk setiap tipe data, dan dalam format berikut.

```
{
    "Execution": {
        "Id": "String",
        "Input": {},
        "Name": "String",
        "RoleArn": "String",
        "StartTime": "Format: ISO 8601",
        "RedriveCount": Number,
        "RedriveTime": "Format: ISO 8601"
    },
    "State": {
        "EnteredTime": "Format: ISO 8601",
        "Name": "String",
        "RetryCount": Number
    },
    "StateMachine": {
        "Id": "String",
        "Name": "String"
    },
    "Task": {
        "Token": "String"
    }
}
```

Selama eksekusi, objek konteks diisi dengan data yang relevan untuk bidang `Parameters` dari asalnya diakses. Nilai untuk bidang `Task` adalah null jika bidang `Parameters` di luar status tugas.

Nilai objek `RedriveCount` konteks adalah 0, jika Anda belum redriveneksekusi. Selanjutnya, objek `RedriveTime` konteks hanya tersedia jika Anda redriven memiliki eksekusi. Jika sudah redriven a Map Run, objek `RedriveTime` konteks hanya tersedia untuk alur kerja anak dari tipe Standard. Untuk redriven Map Run dengan alur kerja turunan tipe Express, `RedriveTime` tidak tersedia.

Konten dari eksekusi yang berjalan termasuk spesifik dalam format berikut.

```
{
    "Execution": {
        "Id": "arn:aws:states:us-
east-1:123456789012:execution:stateMachineName:executionName",
        "Input": {
           "key": "value"
        },
        "Name": "executionName",
        "RoleArn": "arn:aws:iam::123456789012:role...",
        "StartTime": "2019-03-26T20:14:13.192Z"
    },
    "State": {
        "EnteredTime": "2019-03-26T20:14:13.192Z",
        "Name": "Test",
        "RetryCount": 3
    },
    "StateMachine": {
        "Id": "arn:aws:states:us-east-1:123456789012:stateMachine:stateMachineName",
        "Name": "stateMachineName"
    },
    "Task": {
        "Token": "h7XRiCdLtd/83p1E0dMccoxlzFhglsdkzpK9mBVKZsp7d9yrT1W"
    }
}
```

> **Note**
>
> Untuk data objek konteks yang terkait dengan status `Map`, lihat Data Objek Konteks untuk Status Peta.

Mengakses Obyek Konteks

Untuk mengakses objek konteks, pertama tentukan nama parameter dengan menambahkan `.$` sampai akhir, seperti yang Anda lakukan saat memilih input status dengan jalur. Kemudian, untuk mengakses data objek konteks bukan input, tambahkan jalur dengan `$$.`. Ini memberitahu AWS Step Functions untuk menggunakan jalur untuk memilih node dalam objek konteks.

Contoh berikut menunjukkan bagaimana Anda dapat mengakses objek konteks, seperti ID eksekusi, nama, waktu mulai, dan redrive hitungan.

- Mengambil dan meneruskan eksekusi ARN ke layanan hilir
- Akses waktu mulai eksekusi dan nama dalam status Pass
- Akses redrive hitungan eksekusi

Mengambil dan meneruskan eksekusi ARN ke layanan hilir

Contoh status Tugas ini menggunakan jalur untuk mengambil dan meneruskan eksekusi Amazon Resource Name (ARN) ke Amazon Simple Queue Service (Amazon SQS) pesan.

```
{
  "Order Flight Ticket Queue": {
    "Type": "Task",
    "Resource": "arn:aws:states:::sqs:sendMessage",
    "Parameters": {
      "QueueUrl": "https://sqs.us-east-1.amazonaws.com/123456789012/flight-purchase",
      "MessageBody": {
        "From": "YVR",
        "To": "SEA",
        "Execution.$": "$$.Execution.Id"
      }
    },
    "Next": "NEXT_STATE"
  }
}
```

Untuk informasi selengkapnya tentang menggunakan token tugas saat memanggil layanan terpadu, lihat Tunggu Panggilan Balik dengan Token Tugas.

Akses waktu mulai eksekusi dan nama dalam status Pass

```
{
  "Comment": "Accessing context object in a state machine",
  "StartAt": "Get execution context data",
  "States": {
    "Get execution context data": {
      "Type": "Pass",
      "Parameters": {
        "startTime.$": "$$.Execution.StartTime",
        "execName.$": "$$.Execution.Name"
      },
      "ResultPath": "$.executionContext",
```

```
      "End": true
    }
  }
}
```

Akses redrive hitungan eksekusi

Contoh definisi status Tugas berikut menunjukkan bagaimana Anda dapat mengakses redrivehitungan eksekusi.

```
{
  "Type": "Task",
  "Resource": "arn:aws:states:::lambda:invoke",
  "OutputPath": "$.Payload",
  "Parameters": {
    "Payload": {
      "Number.$": "$$.Execution.RedriveCount"
    },
    "FunctionName": "arn:aws:lambda:us-east-2:123456789012:function:functionName"
  },
  "End": true
}
```

Data Objek Konteks untuk Status Peta

Ada dua item tambahan yang tersedia dalam objek konteks ketika memproses status `Map`: `Index` dan `Value`. Untuk setiap iterasi `Map` status, `Index` berisi nomor indeks untuk item array yang sedang diproses, sementara `Value` berisi item array yang sedang diproses. Dalam `Map` keadaan, objek konteks mencakup data berikut:

```
"Map": {
   "Item": {
      "Index": Number,
      "Value": "String"
   }
}
```

Ini hanya tersedia di `Map` negara bagian, dan dapat ditentukan di `ItemSelector` lapangan.

> **Note**
>
> Anda harus menentukan parameter dari objek konteks di blok `ItemSelector` dari status `Map` utama, tidak dalam status yang termasuk dalam bagian `ItemProcessor`.

Mengingat mesin status dengan status `Map` sederhana, kita dapat menyuntikkan informasi dari objek konteks sebagai berikut.

```
{
  "StartAt": "ExampleMapState",
  "States": {
    "ExampleMapState": {
      "Type": "Map",
      "ItemSelector": {
        "ContextIndex.$": "$$.Map.Item.Index",
        "ContextValue.$": "$$.Map.Item.Value"
      },
      "ItemProcessor": {
        "ProcessorConfig": {
          "Mode": "INLINE"
        },
        "StartAt": "TestPass",
        "States": {
          "TestPass": {
            "Type": "Pass",
            "End": true
          }
        }
      },
      "End": true
    }
  }
}
```

Jika Anda mengeksekusi mesin status sebelumnya dengan input berikut, `Index` dan `Value` dimasukkan dalam output.

```
[
  {
    "who": "bob"
```

```
  },
  {
    "who": "meg"
  },
  {
    "who": "joe"
  }
]
```

Output untuk eksekusi mengembalikan nilai `Index` dan `Value` item untuk masing-masing dari tiga iterasi sebagai berikut:

```
[
  {
    "ContextIndex": 0,
    "ContextValue": {
      "who": "bob"
    }
  },
  {
    "ContextIndex": 1,
    "ContextValue": {
      "who": "meg"
    }
  },
  {

    "ContextIndex": 2,
    "ContextValue": {
      "who": "joe"
    }
  }
]
```

Simulator aliran data

Anda dapat merancang, mengimplementasikan, dan men-debug alur kerja di konsol Step Functions. Anda juga dapat mengontrol aliran data dalam alur kerja Anda dengan pemrosesan data JsonPathinput dan output. Dengan simulator aliran Data, Anda dapat mensimulasikan urutan bahwa Tugas status dalam data proses alur kerja Anda saat runtime. Dengan menggunakan simulator, Anda dapat memahami cara memfilter dan memanipulasi data saat mengalir dari satu keadaan ke keadaan

lainnya. Ini mensimulasikan masing-masing bidang berikut yang digunakan Step Functions untuk memproses dan mengontrol aliran data JSON:

`InputPath`

Memilih bagian APA dari seluruh muatan masukan yang akan digunakan sebagai masukan tugas. Jika Anda menentukan bidang ini, Step Functions pertama-tama menerapkan bidang ini.

`Parameter`

Menentukan BAGAIMANA input akan terlihat seperti sebelum memohon tugas. Dengan `Parameters` lapangan, Anda dapat membuat koleksi pasangan kunci-nilai yang dilewatkan sebagai masukan ke Layanan AWSintegrasi, seperti fungsi. AWS Lambda Nilai-nilai ini dapat statis, atau dipilih secara dinamis baik dari input negara atau objek konteks alur kerja.

`ResultSelector`

Menentukan APA untuk memilih dari output tugas. Dengan `ResultSelector` bidang, Anda dapat membuat koleksi pasangan kunci-nilai yang menggantikan hasil negara dan meneruskan koleksi itu. `ResultPath`

`ResultPath`

Menentukan WHERE untuk menempatkan output tugas. Gunakan `ResultPath` untuk menentukan apakah output dari suatu negara adalah salinan dari input, hasil yang dihasilkannya, atau kombinasi keduanya.

`OutputPath`

Menentukan APA yang akan dikirim ke keadaan berikutnya. Dengan `OutputPath` itu, Anda dapat memfilter informasi yang tidak diinginkan, dan hanya meneruskan sebagian data JSON yang Anda pedulikan.

Dalam topik ini:

- Menggunakan simulator aliran data
- Pertimbangan tentang menggunakan simulator aliran Data

Menggunakan simulator aliran data

Simulator menyediakan real-time, side-by-side perbandingan data Anda sebelum dan sesudah Anda menerapkan bidang pemrosesan data input dan output. Untuk menggunakan simulator, tentukan

input JSON. Kemudian, evaluasi melalui masing-masing bidang pemrosesan input dan output. Simulator secara otomatis memvalidasi input JSON Anda dan menyoroti kesalahan sintaks apa pun.

Untuk menggunakan simulator aliran data

Pada langkah-langkah berikut, Anda memberikan masukan JSON dan menerapkan `InputPath` dan `Parameter` bidang. Anda juga dapat menerapkan bidang lain yang tersedia dan melihat output mereka.

1. Buka Konsol Step Functions.
2. Di panel navigasi, pilih Simulator aliran data.
3. Di area input State, ganti contoh data JSON yang telah diisi sebelumnya dengan data JSON berikut. Lalu, pilih Selanjutnya.

```
{
  "data": {
    "firstname": "Jane",
    "lastname": "Doe",
    "identity": {
      "email": "jdoe@example.com",
      "ssn": "123-45-6789"
    },
    "address": {
      "street": "123 Main St",
      "city": "Columbus",
      "state": "OH",
      "zip": "43219"
    }
  }
}
```

4. Untuk InputPath, masukkan **`$.data.address`** untuk memilih node alamat dari input data JSON.

   Masukan Negara setelah InputPath kotak menampilkan hasil berikut.

```
{
  "street": "123 Main St",
  "city": "Columbus",
  "state": "OH",
  "zip": "43219"
```

```
}
```

5. Pilih Selanjutnya.
6. Terapkan `Parameters` bidang untuk mengonversi data JSON yang dihasilkan menjadi string. Untuk mengonversi data, lakukan hal berikut:

   - Di kotak Parameter, masukkan kode berikut untuk membuat string yang disebut`addressString`.

     ```
     {
         "addressString.$": "States.Format('{}. {}, {} - {}', $.street, $.city,
      $.state, $.zip)"
       }
     ```

7. Lihat hasil aplikasi `Parameters` lapangan di Masukan disaring setelah Parameter kotak.

Pertimbangan tentang menggunakan simulator aliran Data

Sebelum Anda menggunakan simulator aliran data, pertimbangkan keterbatasannya, termasuk, namun tidak terbatas pada:

- Ekspresi filter yang tidak didukung

  Ekspresi filter di simulator berperilaku berbeda dari pada layanan Step Functions. Ini termasuk ekspresi yang menggunakan sintaks berikut:. `[?(expression)]` Berikut ini adalah daftar ekspresi yang tidak didukung. Jika digunakan, ekspresi ini mungkin tidak mengembalikan hasil yang diharapkan setelah evaluasi mereka.

  - `$..book[?(@.isInStock==true)]`
  - `$..book[?(@.price > 10.0)].title`

- JsonPathEvaluasi salah untuk array item tunggal

  Jika Anda memfilter data Anda dengan JsonPath ekspresi yang akan mengembalikan item array tunggal, simulator mengembalikan item tanpa array. Sebagai contoh, mempertimbangkan array berikut data, yang disebut`items`:

  ```
  {
    "items": [
      {
  ```

```
      "name": "shoe",
      "color": "blue",
      "comment": "nice shoe"
    },
    {
      "name": "hat",
      "color": "red"
    },
    {
      "name": "shirt",
      "color": "yellow"
    }
  ]
}
```

Mengingat `items` array ini, jika Anda masuk **`$..comment`** di InputPath lapangan, Anda akan mengharapkan output sebagai berikut:

```
[
  "nice shoe"
]
```

Namun, simulator aliran Data mengembalikan output berikut sebagai gantinya:

```
"nice shoe"
```

Untuk JsonPath evaluasi array yang berisi beberapa item, simulator mengembalikan output yang diharapkan.

Mengelola penerapan berkelanjutan dengan versi dan alias

Anda dapat menggunakan Step Functions untuk mengelola penerapan berkelanjutan alur kerja Anda melalui versi mesin status dan alias. Versi adalah snapshot bernomor dan tidak berubah dari mesin negara yang dapat Anda jalankan. Alias adalah pointer hingga dua versi dari mesin negara.

Anda dapat mempertahankan beberapa versi mesin negara Anda dan mengelola penyebarannya dalam alur kerja produksi Anda. Dengan alias, Anda dapat merutekan lalu lintas antara versi alur kerja yang berbeda dan secara bertahap menyebarkan alur kerja tersebut ke lingkungan produksi.

Selain itu, Anda dapat memulai eksekusi mesin negara menggunakan versi atau alias. Jika Anda tidak menggunakan versi atau alias saat memulai eksekusi mesin state, Step Functions menggunakan revisi terbaru dari definisi mesin state.

> **Revisi mesin negara**
>
> Mesin negara dapat memiliki satu atau lebih revisi. Ketika Anda memperbarui mesin negara menggunakan tindakan UpdateStateMachineAPI, itu membuat revisi mesin negara baru. Revisi adalah snapshot hanya-baca yang tidak dapat diubah dari definisi dan konfigurasi mesin negara. Anda tidak dapat memulai eksekusi mesin status dari revisi, dan revisi tidak memiliki ARN. Revisi memilikirevisionId, yang merupakan identifier universal unik (UUID).

Konten

- Versi mesin negara
- Alias mesin negara
- Otorisasi untuk versi dan alias
- Mengaitkan eksekusi mesin status dengan versi atau alias
- Contoh penyebaran alias dan versi
- Lakukan penerapan versi mesin status secara bertahap

Versi mesin negara

Versi adalah snapshot bernomor dan tidak dapat diubah dari mesin negara. Anda mempublikasikan versi dari revisi terbaru yang dibuat untuk mesin negara itu. Setiap versi memiliki Nama Sumber Daya Amazon (ARN) yang unik. ARN ini adalah kombinasi dari ARN mesin keadaan dan nomor versi yang dipisahkan oleh titik dua (:). Contoh berikut menunjukkan format ARN versi mesin negara.

```
arn:partition:states:region:account-id:stateMachine:myStateMachine:1
```

Untuk mulai menggunakan versi mesin negara, Anda harus menerbitkan versi pertama. Setelah memublikasikan versi, Anda dapat menjalankan tindakan StartExecutionAPI dengan versi ARN. Anda tidak dapat mengedit versi, tetapi Anda dapat memperbarui mesin status dan menerbitkan versi baru. Anda juga dapat mempublikasikan beberapa versi mesin negara Anda.

Saat Anda memublikasikan versi baru mesin status Anda, Step Functions memberinya nomor versi. Nomor versi mulai dari 1 dan meningkat secara monoton untuk setiap versi baru. Nomor versi tidak digunakan kembali untuk mesin status tertentu. Jika Anda menghapus versi 10 dari mesin status Anda dan kemudian menerbitkan versi baru, Step Functions menerbitkannya sebagai versi 11.

Properti berikut ini sama untuk semua versi mesin negara:

- Semua versi mesin negara berbagi jenis yang sama (Standar atau Ekspres).
- Anda tidak dapat mengubah nama atau tanggal pembuatan mesin status antar versi.
- Tag berlaku secara global untuk mesin negara. Anda dapat mengelola tag untuk mesin status menggunakan tindakan TagResourcedan UntagResourceAPI.

Mesin negara juga mengandung properti yang merupakan bagian dari setiap versi danrevision, tetapi properti ini dapat berbeda antara dua versi atau revisi yang diberikan. Properti ini termasuk definisi mesin State, peran IAM, konfigurasi penelusuran, dan konfigurasi logging.

Daftar Isi

- Menerbitkan versi mesin status (Konsol)
- Mengelola versi dengan operasi Step Functions API
- Menjalankan versi mesin status dari konsol

Menerbitkan versi mesin status (Konsol)

Anda dapat mempublikasikan hingga 1000 versi mesin negara. Untuk meminta peningkatan batas lunak ini, gunakan halaman Support Center di halaman AWS Management Console. Anda dapat menghapus versi yang tidak digunakan secara manual dari konsol atau dengan menjalankan tindakan DeleteStateMachineVersionAPI.

Untuk mempublikasikan versi mesin negara

1. Buka konsol Step Functions, lalu pilih state machine yang ada.
2. Pada halaman detail mesin Status, pilih Edit.
3. Edit definisi mesin status sesuai kebutuhan, lalu pilih Simpan.
4. Pilih versi Publikasikan.
5. (Opsional) Di bidang Deskripsi pada kotak dialog yang muncul, masukkan deskripsi singkat tentang versi mesin negara.
6. Pilih Terbitkan.

> **Note**
>
> Saat Anda memublikasikan versi baru mesin status Anda, Step Functions memberinya nomor versi. Nomor versi mulai dari 1 dan meningkat secara monoton untuk setiap versi baru. Nomor versi tidak digunakan kembali untuk mesin status tertentu. Jika Anda menghapus versi 10 dari mesin status Anda dan kemudian menerbitkan versi baru, Step Functions menerbitkannya sebagai versi 11.

Mengelola versi dengan operasi Step Functions API

Step Functions menyediakan operasi API berikut untuk mempublikasikan dan mengelola versi mesin status:

- PublishStateMachineVersion— Menerbitkan versi dari mesin revision negara saat ini.
- UpdateStateMachine— Menerbitkan versi mesin status baru jika Anda memperbarui mesin status dan mengatur `publish` parameter ke `true` dalam permintaan yang sama.
- CreateStateMachine— Menerbitkan revisi pertama mesin status jika Anda mengatur `publish` parameter ke. `true`
- ListStateMachineVersions— Daftar versi untuk ARN mesin negara yang ditentukan.
- DescribeStateMachine— Mengembalikan detail versi mesin negara untuk versi ARN yang ditentukan dalam. `stateMachineArn`
- DeleteStateMachineVersion— Menghapus versi mesin negara.

Untuk mempublikasikan versi baru dari revisi saat ini dari mesin status yang disebut *myStateMachine* menggunakan AWS Command Line Interface, gunakan `publish-state-machine-version` perintah:

```
aws stepfunctions publish-state-machine-version --state-machine-arn arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine
```

Respons mengembalikan`stateMachineVersionArn`. Misalnya, perintah sebelumnya mengembalikan respon dari`arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:1`.

> **Note**
>
> Saat Anda memublikasikan versi baru mesin status Anda, Step Functions memberinya nomor versi. Nomor versi mulai dari 1 dan meningkat secara monoton untuk setiap versi baru. Nomor versi tidak digunakan kembali untuk mesin status tertentu. Jika Anda menghapus versi 10 dari mesin status Anda dan kemudian menerbitkan versi baru, Step Functions menerbitkannya sebagai versi 11.

Menjalankan versi mesin status dari konsol

Untuk mulai menggunakan versi mesin status, Anda harus terlebih dahulu mempublikasikan versi dari mesin status saat inirevision. Untuk memublikasikan versi, gunakan konsol Step Functions atau jalankan tindakan PublishStateMachineVersionAPI. Anda juga dapat menjalankan tindakan UpdateStateMachineAliasAPI dengan parameter opsional bernama `publish` untuk memperbarui mesin status dan mempublikasikan versinya.

Anda dapat memulai eksekusi versi dengan menggunakan konsol atau dengan menjalankan tindakan StartExecutionAPI dan menyediakan versi ARN. Anda juga dapat menggunakan alias untuk memulai eksekusi versi. Berdasarkan konfigurasi routing-nya, alias merutekan lalu lintas ke versi tertentu.

Jika Anda memulai eksekusi mesin status tanpa menggunakan versi, Step Functions menggunakan revisi terbaru dari mesin status untuk eksekusi. Untuk informasi tentang cara Step Functions mengaitkan eksekusi dengan versi, lihatMengaitkan eksekusi dengan versi atau alias.

Untuk memulai eksekusi menggunakan versi mesin negara

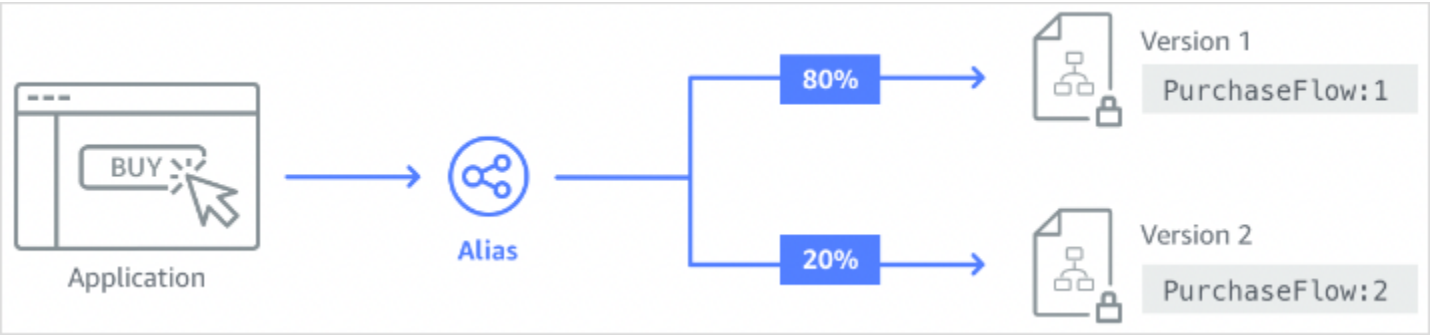
1. Buka konsol Step Functions, lalu pilih mesin status yang sudah ada yang telah Anda terbitkan untuk satu atau beberapa versi. Untuk mempelajari cara mempublikasikan versi, lihatMenerbitkan versi mesin status (Konsol).
2. Pada halaman detail mesin Status, pilih tab Versi.
3. Di bagian Versi, lakukan hal berikut:
   a. Pilih versi yang ingin Anda mulai eksekusi.
   b. Pilih Mulai Eksekusi.
4. (Opsional) Dalam Mulai eksekusi kotak dialog, masukkan nama untuk eksekusi.
5. (Opsional), masukkan input eksekusi, lalu pilih Mulai eksekusi.

Alias mesin negara

Alias adalah pointer hingga dua versi dari mesin state yang sama. Anda dapat membuat beberapa alias untuk mesin negara Anda. Setiap alias memiliki Nama Sumber Daya Amazon (ARN) yang unik. Alias ARN adalah kombinasi dari ARN mesin negara dan nama alias, dipisahkan oleh titik dua (:). Contoh berikut menunjukkan format mesin negara alias ARN.

```
arn:partition:states:region:account-id:stateMachine:myStateMachine:aliasName
```

Anda dapat menggunakan alias untuk merutekan lalu lintas antara salah satu dari dua versi mesin negara. Anda juga dapat membuat alias yang menunjuk ke satu versi. Alias hanya dapat menunjuk ke versi mesin status. Anda tidak dapat menggunakan alias untuk menunjuk ke alias lain. Anda juga dapat memperbarui alias untuk menunjuk ke versi mesin status yang berbeda.



Daftar Isi

- Membuat alias mesin status (Konsol)

- Mengelola alias dengan operasi Step Functions API
- Konfigurasi perutean alias
- Menjalankan mesin status menggunakan alias (Konsol)

Membuat alias mesin status (Konsol)

Anda dapat membuat hingga 100 alias untuk setiap state machine dengan menggunakan konsol Step Functions atau dengan menjalankan aksi CreateStateMachineAliasAPI. Untuk meminta peningkatan batas lunak ini, gunakan halaman Support Center di halaman AWS Management Console. Hapus alias yang tidak digunakan dari konsol atau dengan menjalankan tindakan API. DeleteStateMachineAlias

Untuk membuat alias mesin negara

1. Buka konsol Step Functions, lalu pilih state machine yang ada.
2. Pada halaman detail mesin State, pilih tab Alias.
3. Pilih Buat alias baru.
4. Di halaman Buat alias, lakukan hal berikut:
   a. Masukkan nama Alias.
   b. (Opsional) Masukkan Deskripsi untuk alias.
5. Untuk mengonfigurasi perutean pada alias, lihat Konfigurasi perutean Alias.
6. Pilih Buat alias.

Mengelola alias dengan operasi Step Functions API

Step Functions menyediakan operasi API berikut yang dapat Anda gunakan untuk membuat dan mengelola alias mesin status atau mendapatkan informasi tentang alias:

- CreateStateMachineAlias— Membuat alias untuk mesin negara.
- DescribeStateMachineAlias— Mengembalikan rincian tentang alias mesin negara.
- ListStateMachineAliases— Daftar alias untuk ARN mesin negara yang ditentukan.
- UpdateStateMachineAlias— Memperbarui konfigurasi alias mesin status yang ada dengan memodifikasi atau. `description` `routingConfiguration`
- DeleteStateMachineAlias— Menghapus versi mesin negara.

Untuk membuat alias bernama *PROD* yang menunjuk ke versi 1 dari mesin negara bernama *myStateMachine* menggunakan AWS Command Line Interface, gunakan `create-state-machine-alias` perintah.

```
aws stepfunctions create-state-machine-alias --name PROD  --routing-
configuration "[{\"stateMachineVersionArn\":\"arn:aws:states:us-
east-1:123456789012:stateMachine:myStateMachine:1\",\"weight\":100}]"
```

Konfigurasi perutean alias

Anda dapat menggunakan alias untuk merutekan lalu lintas eksekusi antara dua versi mesin negara. Misalnya, Anda ingin meluncurkan versi baru dari mesin status Anda. Anda dapat mengurangi risiko yang terlibat dalam penerapan versi baru dengan mengonfigurasi perutean pada alias. Dengan mengonfigurasi perutean, Anda dapat mengirim sebagian besar lalu lintas Anda ke versi mesin status Anda yang telah diuji sebelumnya. Versi baru kemudian dapat menerima persentase yang lebih kecil, sampai Anda dapat mengonfirmasi bahwa aman untuk meneruskan versi baru.

Untuk menentukan konfigurasi perutean, pastikan Anda memublikasikan kedua versi mesin status yang ditunjuk alias Anda. Ketika Anda memulai eksekusi dari alias, Step Functions secara acak memilih versi state machine untuk dijalankan dari versi yang ditentukan dalam konfigurasi routing. Ini mendasarkan pilihan ini pada persentase lalu lintas yang Anda tetapkan untuk setiap versi dalam konfigurasi routing alias.

Untuk mengkonfigurasi konfigurasi routing pada alias

- Pada halaman Create alias, di bawah konfigurasi Routing, lakukan hal berikut:
  a. Untuk Versi, pilih versi mesin status pertama yang ditunjuk alias.
  b. Pilih kotak centang Pisahkan lalu lintas antara dua versi.

  > **Tip**
  >
  > Untuk menunjuk ke satu versi, kosongkan kotak centang Split traffic antara dua versi.

  c. Untuk Versi, pilih versi kedua yang harus ditunjukkan oleh alias.

d. Di bidang Persentase lalu lintas, tentukan persentase lalu lintas untuk merutekan ke setiap versi. Misalnya, masukkan **60** dan **40** rute 60 persen lalu lintas eksekusi ke versi pertama dan 40 persen lalu lintas ke versi kedua.

   Persentase lalu lintas gabungan harus sama dengan 100 persen.

Menjalankan mesin status menggunakan alias (Konsol)

Anda dapat memulai eksekusi mesin status dengan alias baik dari konsol atau dengan menjalankan aksi StartExecutionAPI dengan ARN alias. Step Functions kemudian menjalankan versi yang ditentukan oleh alias. Secara default, jika Anda tidak menentukan versi atau alias saat memulai eksekusi mesin status, Step Functions menggunakan revisi terbaru.

Untuk memulai eksekusi mesin status menggunakan alias

1. Buka konsol Step Functions, lalu pilih mesin status yang sudah ada yang telah Anda buat alias. Untuk informasi tentang membuat alias, lihatMembuat alias mesin status (Konsol).
2. Pada halaman detail mesin State, pilih tab Alias.
3. Di bagian Alias, lakukan hal berikut:
   a. Pilih alias yang ingin Anda gunakan untuk memulai eksekusi.
   b. Pilih Mulai Eksekusi.
4. (Opsional) Dalam Mulai eksekusi kotak dialog, masukkan nama untuk eksekusi.
5. Jika diperlukan, masukkan input eksekusi, lalu pilih Mulai eksekusi.

Otorisasi untuk versi dan alias

Untuk menjalankan tindakan API Step Functions dengan versi atau alias, Anda memerlukan izin yang sesuai. Untuk mengotorisasi versi atau alias untuk memanggil tindakan API, Step Functions menggunakan ARN mesin state alih-alih menggunakan ARN versi atau alias ARN. Anda juga dapat menurunkan izin untuk versi atau alias tertentu. Untuk informasi selengkapnya, lihat Melingkupi izin.

Anda dapat menggunakan contoh kebijakan IAM berikut dari mesin negara bernama *myStateMachine* untuk memanggil tindakan CreateStateMachineAliasAPI untuk membuat alias mesin negara.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "states:CreateStateMachineAlias",
      "Resource": "arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine"
    }
  ]
}
```

Saat Anda menyetel izin untuk mengizinkan atau menolak akses ke tindakan API menggunakan versi atau alias mesin status, pertimbangkan hal berikut:

- Jika Anda menggunakan `publish` parameter tindakan CreateStateMachinedan UpdateStateMachineAPI untuk mempublikasikan versi mesin status baru, Anda juga memerlukan `ALLOW` izin pada tindakan PublishStateMachineVersionAPI.
- Tindakan DeleteStateMachineAPI menghapus semua versi dan alias yang terkait dengan mesin negara.

Dalam topik ini:

- Melingkupi izin untuk versi atau alias

Melingkupi izin untuk versi atau alias

Anda dapat menggunakan qualifier untuk lebih jauh cakupan bawah izin otorisasi yang diperlukan oleh versi atau alias. Kualifikasi mengacu pada nomor versi atau nama alias. Anda menggunakan kualifikasi untuk memenuhi syarat mesin negara. Contoh berikut adalah mesin negara ARN yang menggunakan alias bernama `PROD` sebagai qualifier.

```
arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:PROD
```

Untuk informasi lebih lanjut tentang ARN yang memenuhi syarat dan tidak memenuhi syarat, lihat. Mengaitkan eksekusi dengan versi atau alias

Anda cakupan bawah izin menggunakan kunci konteks opsional bernama `states:StateMachineQualifier` dalam pernyataan kebijakan IAM. `Condition` Misalnya, kebijakan IAM berikut untuk mesin negara bernama `myStateMachine` menolak akses untuk memanggil tindakan DescribeStateMachineAPI dengan alias bernama sebagai `PROD` atau versi. 1

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "states:DescribeStateMachine",
      "Resource": "arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "states:StateMachineQualifier": [
            "PROD",
            "1"
          ]
        }
      }
    }
  ]
}
```

Daftar berikut menentukan tindakan API di mana Anda dapat cakupan bawah izin dengan kunci `StateMachineQualifier` konteks.

- CreateStateMachineAlias
- DeleteStateMachineAlias
- DeleteStateMachineVersion
- DescribeStateMachine
- DescribeStateMachineAlias
- ListExecutions
- ListStateMachineAliases
- StartExecution
- StartSyncExecution
- UpdateStateMachineAlias

Mengaitkan eksekusi mesin status dengan versi atau alias

Step Functions mengaitkan eksekusi dengan versi atau alias berdasarkan Amazon Resource Name (ARN) yang Anda gunakan untuk menjalankan tindakan API. StartExecution Step Functions melakukan tindakan ini pada waktu mulai eksekusi.

Anda dapat memulai eksekusi mesin negara menggunakan ARN yang memenuhi syarat atau tidak memenuhi syarat.

- ARN yang memenuhi syarat - Mengacu pada ARN mesin negara yang diakhiran dengan nomor versi atau nama alias.

  Contoh ARN yang memenuhi syarat berikut mengacu pada 3 versi mesin negara bernama. `myStateMachine`

  ```
  arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:3
  ```

  Contoh ARN yang memenuhi syarat berikut mengacu pada alias bernama mesin PROD negara bernama. `myStateMachine`

  ```
  arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:PROD
  ```

- ARN yang tidak memenuhi syarat - Mengacu pada ARN mesin negara tanpa nomor versi atau akhiran nama alias.

  ```
  arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine
  ```

Misalnya, jika ARN Anda yang memenuhi syarat mengacu pada versi3, Step Functions mengaitkan eksekusi dengan versi ini. Itu tidak mengaitkan eksekusi dengan alias apa pun yang mengarah ke versi3.

Jika ARN Anda yang memenuhi syarat mengacu pada alias, Step Functions mengaitkan eksekusi dengan alias tersebut dan versi yang ditunjuk alias tersebut. Eksekusi hanya dapat dikaitkan dengan satu alias.

> **Note**
>
> Jika Anda memulai eksekusi dengan ARN yang tidak memenuhi syarat, Step Functions tidak mengaitkan eksekusi tersebut dengan versi meskipun versi tersebut menggunakan

mesin status yang sama. revision Misalnya, jika versi 3 menggunakan revisi terbaru, tetapi Anda memulai eksekusi dengan ARN yang tidak memenuhi syarat, Step Functions tidak mengaitkan eksekusi tersebut dengan versi 3.

Dalam topik ini:

- Melihat eksekusi dimulai dengan versi atau alias

Melihat eksekusi dimulai dengan versi atau alias

Step Functions menyediakan cara-cara berikut di mana Anda dapat melihat eksekusi yang dimulai dengan versi atau alias:

Menggunakan tindakan API

Anda dapat melihat semua eksekusi yang terkait dengan versi atau alias dengan menjalankan tindakan API DescribeExecutiondan ListExecutions. Tindakan API ini mengembalikan ARN versi atau alias yang digunakan untuk memulai eksekusi. Tindakan ini juga mengembalikan detail lainnya termasuk status dan ARN eksekusi.

Anda juga dapat menyediakan mesin status alias ARN atau versi ARN untuk mencantumkan eksekusi yang terkait dengan alias atau versi tertentu.

Contoh respons tindakan ListExecutionsAPI berikut menunjukkan ARN alias yang digunakan untuk memulai eksekusi mesin status bernama. *myFirstExecution*

Teks yang *dicetak miring* dalam cuplikan kode berikut mewakili informasi khusus sumber daya.

```
{
    "executions": [
        {
            "executionArn": "arn:aws:states:us-
east-1:123456789012:execution:myStateMachine:myFirstExecution",
            "stateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:myStateMachine",
            "stateMachineAliasArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:myStateMachine:PROD",
            "name": "myFirstExecution",
            "status": "SUCCEEDED",
```

```
            "startDate": "2023-04-20T23:07:09.477000+00:00",
            "stopDate": "2023-04-20T23:07:09.732000+00:00"
        }
    ]
}
```

Menggunakan konsol Step Functions

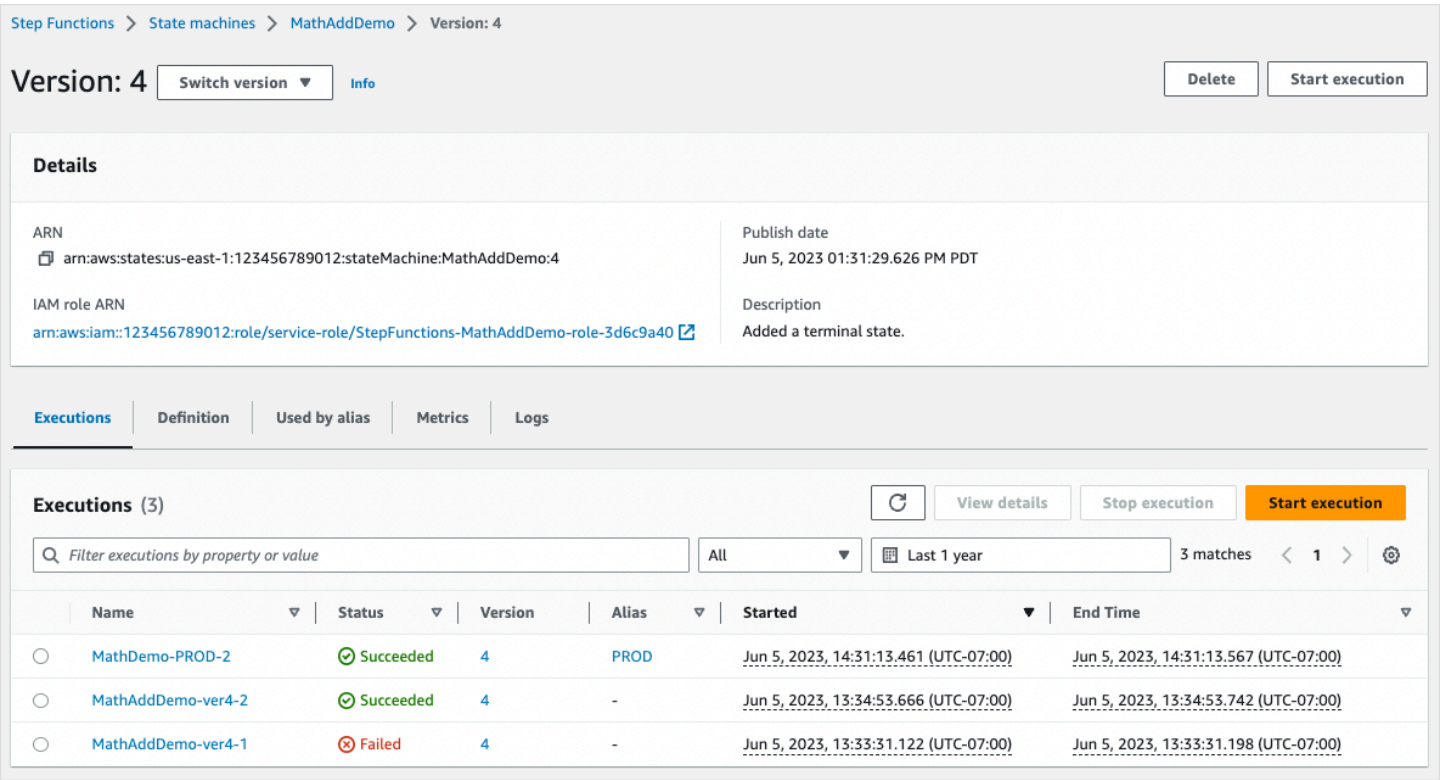
Anda juga dapat melihat eksekusi yang dimulai oleh versi atau alias dari konsol Step Functions. Prosedur berikut menunjukkan bagaimana Anda dapat melihat eksekusi yang dimulai dengan versi tertentu:

1. Buka konsol Step Functions, lalu pilih mesin status yang sudah ada yang telah Anda publikasikan versinya atau buat alias. Contoh ini menunjukkan cara melihat eksekusi yang dimulai dengan versi mesin status tertentu.
2. Pilih tab Versi, lalu pilih versi dari daftar Versi.

   > **Tip**
   >
   > Filter berdasarkan properti atau kotak nilai untuk mencari versi tertentu.

3. Pada halaman Detail versi, Anda dapat melihat daftar semua eksekusi mesin status yang sedang berlangsung dan sebelumnya yang dimulai dengan versi yang dipilih.

Gambar berikut menunjukkan halaman konsol Detail Versi. Halaman ini mencantumkan eksekusi yang dimulai oleh versi 4 dari mesin negara bernama*MathAddDemo*. Daftar ini juga menampilkan eksekusi yang dimulai oleh alias bernama*PROD*. Alias ini mengarahkan lalu lintas eksekusi ke versi 4.



Menggunakan CloudWatch metrik

Untuk setiap eksekusi mesin status yang Anda mulai dengan[Qualified ARN](), Step Functions memancarkan metrik tambahan dengan nama dan nilai yang sama dengan metrik yang dipancarkan saat ini. Metrik tambahan ini berisi dimensi untuk masing-masing pengenal versi dan nama alias yang Anda gunakan untuk memulai eksekusi. Dengan metrik ini, Anda dapat memantau eksekusi mesin status pada tingkat versi dan menentukan kapan skenario rollback mungkin diperlukan. Anda juga dapat [membuat CloudWatch alarm Amazon]() berdasarkan metrik ini.

Step Functions memancarkan metrik berikut untuk eksekusi yang Anda mulai dengan alias atau versi:

- `ExecutionTime`
- `ExecutionsAborted`
- `ExecutionsFailed`
- `ExecutionsStarted`
- `ExecutionsSucceeded`
- `ExecutionsTimedOut`

Jika Anda memulai eksekusi dengan versi ARN, Step Functions menerbitkan metrik dengan dan metrik kedua dengan `StateMachineArn` `StateMachineArn` dan dimensi. `Version`

Jika Anda memulai eksekusi dengan alias ARN, Step Functions memancarkan metrik berikut:

- Dua metrik untuk ARN dan versi yang tidak memenuhi syarat.
- Sebuah metrik dengan `StateMachineArn` dan `Alias` dimensi.

Contoh penyebaran alias dan versi

Contoh teknik penyebaran Canary berikut menunjukkan bagaimana Anda dapat menerapkan versi mesin status baru dengan. AWS Command Line Interface Dalam contoh ini, alias Anda membuat rute 20 persen dari lalu lintas eksekusi ke versi baru. Kemudian rute sisanya 80 persen versi sebelumnya. Untuk menerapkan versi mesin status baru dan menggeser lalu lintas eksekusi dengan alias, selesaikan langkah-langkah berikut:

1. Publikasikan versi dari revisi mesin keadaan saat ini.

   Gunakan publish-state-machine-version perintah di AWS CLI untuk mempublikasikan versi dari revisi saat ini dari mesin negara yang disebut *myStateMachine*:

   ```
   aws stepfunctions publish-state-machine-version --state-machine-arn
    arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine
   ```

   Respon mengembalikan `stateMachineVersionArn` versi yang Anda terbitkan. Sebagai contoh, `arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:1`.
2. Buat alias yang menunjuk ke versi mesin negara.

   Gunakan create-state-machine-alias perintah untuk membuat alias bernama *PROD* yang menunjuk ke versi 1 dari*myStateMachine*:

   ```
   aws stepfunctions create-state-machine-alias --name PROD --routing-
   configuration "[{\"stateMachineVersionArn\":\"arn:aws:states:us-
   east-1:123456789012:stateMachine:myStateMachine:1\",\"weight\":100}]"
   ```
3. Verifikasi bahwa eksekusi yang dimulai oleh alias menggunakan versi publikasi yang benar.

   Mulai eksekusi baru *myStateMachine* dengan menyediakan ARN alias **PROD** dalam perintahstart-execution:

```
aws stepfunctions start-execution
  --state-machine-arn arn:aws:states:us-
east-1:123456789012:stateMachineAlias:myStateMachine:PROD
  --input "{}"
```

Jika Anda menyediakan mesin negara ARN dalam StartExecutionpermintaan, menggunakan terbaru revision dari mesin negara bukan versi yang ditentukan dalam alias Anda untuk memulai eksekusi.

4. Perbarui definisi mesin negara dan publikasikan versi baru.

   Perbarui *myStateMachine* dan publikasikan versi barunya. Untuk melakukan ini, gunakan `publish` parameter opsional dari update-state-machine perintah:

   ```
   aws stepfunctions update-state-machine
       --state-machine-arn arn:aws:states:us-
   east-1:123456789012:stateMachine:myStateMachine
       --definition $UPDATED_STATE_MACHINE_DEFINITION
       --publish
   ```

   Respon mengembalikan `stateMachineVersionArn` untuk versi baru. Sebagai contoh, `arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:2`.

5. Perbarui alias untuk menunjuk ke kedua versi dan mengatur konfigurasi perutean alias.

   Gunakan update-state-machine-alias perintah untuk memperbarui konfigurasi routing `PROD` alias. Konfigurasikan alias sehingga 80 persen dari lalu lintas eksekusi masuk ke versi 1 dan 20 persen sisanya masuk ke versi 2:

   ```
   aws stepfunctions update-state-machine-alias --state-machine-alias-arn
    arn:aws:states:us-east-1:123456789012:stateMachineAlias:myStateMachine:PROD
    --routing-configuration "[{\"stateMachineVersionArn\":
   \"arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:1\",
   \"weight\":80}, {\"stateMachineVersionArn\":\"arn:aws:states:us-
   east-1:123456789012:stateMachine:myStateMachine:2\",\"weight\":20}]"
   ```

6. Ganti versi 1 dengan versi 2.

   Setelah Anda memverifikasi bahwa versi mesin status baru Anda berfungsi dengan benar, Anda dapat menerapkan versi mesin status baru. Untuk melakukan ini, perbarui alias lagi untuk menetapkan 100 persen lalu lintas eksekusi ke versi baru.

Gunakan update-state-machine-alias perintah untuk mengatur konfigurasi routing alias menjadi 100 persen PROD untuk versi 2:

```
aws stepfunctions update-state-machine-alias --state-machine-alias-arn
 arn:aws:states:us-east-1:123456789012:stateMachineAlias:myStateMachine:PROD
 --routing-configuration "[{\"stateMachineVersionArn\":\"arn:aws:states:us-
east-1:123456789012:stateMachine:myStateMachine:2\",\"weight\":100}]"
```

Tip

Untuk memutar kembali penyebaran versi 2, edit konfigurasi perutean alias untuk menggeser 100 persen lalu lintas ke versi yang baru digunakan.

```
aws stepfunctions update-state-machine-alias
  --state-machine-alias-arn arn:aws:states:us-
east-1:123456789012:stateMachineAlias:myStateMachine:PROD
  --routing-configuration "[{\"stateMachineVersionArn\":\"arn:aws:states:us-
east-1:123456789012:stateMachine:myStateMachine:1\",\"weight\":100}]"
```

Anda dapat menggunakan versi dan alias untuk melakukan jenis penyebaran lainnya. Misalnya, Anda dapat melakukan penyebaran bergulir versi baru mesin negara Anda. Untuk melakukannya, secara bertahap meningkatkan persentase tertimbang dalam konfigurasi routing alias yang menunjuk ke versi baru.

Anda juga dapat menggunakan versi dan alias untuk melakukan penyebaran biru/hijau. Untuk melakukannya, buat alias bernama `green` yang menjalankan versi 1 saat ini dari mesin negara Anda. Kemudian, buat alias lain bernama `blue` yang menjalankan versi baru, misalnya,*2*. Untuk menguji versi baru, kirim lalu lintas eksekusi ke `blue` alias. Ketika Anda yakin bahwa versi baru Anda berfungsi dengan benar, perbarui `green` alias untuk menunjuk ke versi baru Anda.

Lakukan penerapan versi mesin status secara bertahap

Penyebaran bergulir adalah strategi penyebaran yang perlahan-lahan menggantikan versi aplikasi sebelumnya dengan versi aplikasi baru. Untuk melakukan penyebaran bergulir dari versi mesin negara, secara bertahap mengirim peningkatan jumlah lalu lintas eksekusi ke versi baru. Jumlah lalu lintas dan tingkat kenaikan adalah parameter yang Anda konfigurasikan.

Anda dapat melakukan penyebaran versi bergulir menggunakan salah satu opsi berikut:

- Langkah Fungsi konsol- Buat alias yang menunjuk ke dua versi mesin negara yang sama. Untuk alias ini, Anda mengonfigurasi konfigurasi perutean untuk menggeser lalu lintas antara dua versi. Untuk informasi selengkapnya tentang menggunakan konsol untuk meluncurkan versi, lihatVersidanAlias.
- Script untukAWS CLIdan SDK- Buat skrip shell menggunakanAWS CLIatauAWSSDK. Untuk informasi selengkapnya, lihat bagian berikut untuk menggunakanAWS CLIdanAWSSDK.
- AWS CloudFormationtemplat— Gunakan`AWS::StepFunctions::StateMachineVersion`dan`AWS::StepFunctions::StateMachi` daya untuk mempublikasikan beberapa versi mesin negara dan membuat alias untuk menunjuk ke satu atau dua versi ini.

GunakanAWS CLIuntuk menyebarkan versi mesin state baru

Contoh skrip di bagian ini menunjukkan bagaimana Anda dapat menggunakanAWS CLIuntuk secara bertahap menggeser lalu lintas dari versi mesin negara sebelumnya ke versi mesin negara baru. Anda dapat menggunakan skrip contoh ini atau memperbaruinya sesuai dengan kebutuhan Anda.

Skrip ini menunjukkan penyebaran Canary untuk menerapkan versi mesin negara baru menggunakan alias. Langkah-langkah berikut menguraikan tugas yang dilakukan skrip:

1. Jika`publish_revision`parameter diatur ke true, mempublikasikan terbarurevisionsebagai versi berikutnya dari mesin negara. Versi ini menjadi versi live baru jika deployment berhasil.

   Jika Anda mengatur`publish_revision`parameter ke false, script menyebarkan versi diterbitkan terakhir dari mesin negara.
2. Buat alias jika belum ada. Jika alias tidak ada, arahkan 100 persen lalu lintas untuk alias ini ke versi baru, dan kemudian keluar dari skrip.
3. Perbarui konfigurasi perutean alias untuk menggeser persentase kecil lalu lintas dari versi sebelumnya ke versi baru. Anda mengatur persentase kenari ini dengan`canary_percentage`parameter.
4. Secara default, pantau yang dapat dikonfigurasiCloudWatchalarm setiap 60 detik. Jika salah satu alarm ini berangkat, kembalikan penyebaran segera dengan mengarahkan 100 persen lalu lintas ke versi sebelumnya.

Setelah setiap interval waktu, dalam hitungan detik, didefinisikan dalam`alarm_polling_interval`, terus pantau alarm. Lanjutkan pemantauan hingga interval waktu yang ditentukan`canary_interval_seconds`telah berlalu.

5. Jika tidak ada alarm yang dimatikan selama`canary_interval_seconds`, menggeser 100 persen lalu lintas ke versi baru.
6. Jika versi baru berhasil diterapkan, hapus versi apa pun yang lebih lama dari nomor yang ditentukan dalam`history_max`parameter.

```
#!/bin/bash
#
# AWS StepFunctions example showing how to create a canary deployment with a
# State Machine Alias and versions.
#
# Requirements: AWS CLI installed and credentials configured.
#
# A canary deployment deploys the new version alongside the old version, while
# routing only a small fraction of the overall traffic to the new version to
# see if there are any errors. Only once the new version has cleared a testing
# period will it start receiving 100% of traffic.
#
# For a Blue/Green or All at Once style deployment, you can set the
# canary_percentage to 100. The script will immediately shift 100% of traffic
# to the new version, but keep on monitoring the alarms (if any) during the
# canary_interval_seconds time interval. If any alarms raise during this period,
# the script will automatically rollback to the previous version.
#
# Step Functions allows you to keep a maximum of 1000 versions in version history
# for a state machine. This script has a version history deletion mechanism at
# the end, where it will delete any versions older than the limit specified.
#
# For a fuller example, that also demonstrates linear (or rolling) deployments,
# please see
# https://github.com/aws-samples/aws-stepfunctions-examples/blob/main/gradual-deploy/
sfndeploy.py

set -euo pipefail

# ************************************************************************
# you can safely change the variables in this block to your values
```

```
state_machine_name="my-state-machine"
alias_name="alias-1"
region="us-east-1"

# array of cloudwatch alarms to poll during the test period.
# to disable alarm checking, set alarm_names=()
alarm_names=("alarm1" "alarm name with a space")

# true to publish the current revision as the next version before deploy.
# false to deploy the latest version from the state machine's version history.
publish_revision=true

# true to force routing configuration update even if the current routing
# for the alias does not have a 100% routing config.
# false will abandon deploy attempt if current routing config not 100% to a
# single version.
# Be careful when you combine this flag with publish_revision - if you just
# rerun the script you might deploy the newly published revision from the
# previous run.
force=false

# percentage of traffic to route to the new version during the test period
canary_percentage=10

# how many seconds the canary deployment lasts before full deploy to 100%
canary_interval_seconds=300

# how often to poll the alarms
alarm_polling_interval=60

# how many versions to keep in history. delete versions prior to this.
# set to 0 to disable old version history deletion.
history_max=0
# ******************************************************************************

######################################
# Update alias routing configuration.
#
# If you don't specify version 2 details, will only create 1 routing entry. In
# this case the routing entry weight must be 100.
#
# Globals:
#   alias_arn
# Arguments:
```

```
#   1. version 1 arn
#   2. version 1 weight
#   3. version 2 arn (optional)
#   4. version 2 weight (optional)
#####################################
function update_routing() {
  if [[ $# -eq 2 ]]; then
    local routing_config="[{\"stateMachineVersionArn\": \"$1\", \"weight\":$2}]"
  elif [[ $# -eq 4 ]]; then
    local routing_config="[{\"stateMachineVersionArn\": \"$1\", \"weight\":$2},
 {\"stateMachineVersionArn\": \"$3\", \"weight\":$4}]"
  else
    echo "You have to call update_routing with either 2 or 4 input arguments." >&2
    exit 1
  fi

  ${aws} update-state-machine-alias --state-machine-alias-arn ${alias_arn} --routing-
configuration "${routing_config}"
}

# ******************************************************************************
# pre-run validation
if [[ (("${#alarm_names[@]}" -gt 0)) ]]; then
  alarm_exists_count=$(aws cloudwatch describe-alarms --alarm-names "${alarm_names[@]}"
 --alarm-types "CompositeAlarm" "MetricAlarm" --query "length([MetricAlarms,
 CompositeAlarms][])" --output text)

  if [[ (("${#alarm_names[@]}" -ne "${alarm_exists_count}")) ]]; then
    echo All of the alarms to monitor do not exist in CloudWatch: $(IFS=,; echo
 "${alarm_names[*]}") >&2
    echo Only the following alarm names exist in CloudWatch:
    aws cloudwatch describe-alarms --alarm-names "${alarm_names[@]}" --alarm-types
 "CompositeAlarm" "MetricAlarm" --query "join(', ', [MetricAlarms, CompositeAlarms]
[].AlarmName)" --output text
    exit 1
  fi
fi

if [[ (("${history_max}" -gt 0)) && (("${history_max}" -lt 2)) ]]; then
  echo The minimum value for history_max is 2. This is the minimum number of older
 state machine versions to be able to rollback in the future. >&2
  exit 1
fi
# ******************************************************************************
```

```
# main block follows

account_id=$(aws sts get-caller-identity --query Account --output text)

sm_arn="arn:aws:states:${region}:${account_id}:stateMachine:${state_machine_name}"

# the aws command we'll be invoking a lot throughout.
aws="aws stepfunctions"

# promote the latest revision to the next version
if [[ "${publish_revision}" = true ]]; then
  new_version=$(${aws} publish-state-machine-version --state-machine-arn=$sm_arn --
query stateMachineVersionArn --output text)
  echo Published the current revision of state machine as the next version with arn:
 ${new_version}
else
  new_version=$(${aws} list-state-machine-versions --state-machine-arn ${sm_arn} --max-
results 1 --query "stateMachineVersions[0].stateMachineVersionArn" --output text)
  echo "Since publish_revision is false, using the latest version from the state
 machine's version history: ${new_version}"
fi

# find the alias if it exists
alias_arn_expected="${sm_arn}:${alias_name}"
alias_arn=$(${aws} list-state-machine-aliases --state-machine-arn
 ${sm_arn} --query "stateMachineAliases[?stateMachineAliasArn==\`
${alias_arn_expected}\`].stateMachineAliasArn" --output text)

if [[ "${alias_arn_expected}" == "${alias_arn}" ]]; then
  echo Found alias ${alias_arn}

  echo Current routing configuration is:
  ${aws} describe-state-machine-alias --state-machine-alias-arn "${alias_arn}" --query
 routingConfiguration
else
  echo Alias does not exist. Creating alias ${alias_arn_expected} and routing 100%
 traffic to new version ${new_version}

  ${aws} create-state-machine-alias --name "${alias_name}" --routing-configuration
 "[{\"stateMachineVersionArn\": \"${new_version}\", \"weight\":100}]"

  echo Done!
  exit 0
fi
```

```
# find the version to which the alias currently points (the current live version)
old_version=$(${aws} describe-state-machine-alias --state-machine-alias-arn $alias_arn
 --query "routingConfiguration[?weight==\`100\`].stateMachineVersionArn" --output text)

if [[ -z "${old_version}" ]]; then
  if [[ "${force}" = true ]]; then
    echo Force setting is true. Will force update to routing config for alias to point
 100% to new version.
    update_routing "${new_version}" 100

    echo Alias ${alias_arn} now pointing 100% to ${new_version}.
    echo Done!
    exit 0
  else
    echo Alias ${alias_arn} does not have a routing config entry with 100% of the
 traffic. This means there might be a deploy in progress, so not starting another
 deploy at this time. >&2
    exit 1
  fi
fi

if [[ "${old_version}" == "${new_version}" ]]; then
  echo The alias already points to this version. No update necessary.
  exit 0
fi

echo Switching ${canary_percentage}% to new version ${new_version}
(( old_weight = 100 - ${canary_percentage} ))
update_routing "${new_version}" ${canary_percentage} "${old_version}" ${old_weight}

echo New version receiving ${canary_percentage}% of traffic.
echo Old version ${old_version} is still receiving ${old_weight}%.

if [[ ${#alarm_names[@]} -eq 0 ]]; then
  echo No alarm_names set. Skipping cloudwatch monitoring.
  echo Will sleep for ${canary_interval_seconds} seconds before routing 100% to new
 version.
  sleep ${canary_interval_seconds}
  echo Canary period complete. Switching 100% of traffic to new version...
else
  echo Checking if alarms fire for the next ${canary_interval_seconds} seconds.

  (( total_wait = canary_interval_seconds + $(date +%s) ))
```

```
  now=$(date +%s)
  while [[ (($${now} -lt ${total_wait})) ]]; do
    alarm_result=$(aws cloudwatch describe-alarms --alarm-names "${alarm_names[@]}"
 --state-value ALARM --alarm-types "CompositeAlarm" "MetricAlarm" --query "join(', ',
 [MetricAlarms, CompositeAlarms][].AlarmName)" --output text)

    if [[ ! -z "${alarm_result}" ]]; then
      echo The following alarms are in ALARM state: ${alarm_result}. Rolling back
 deploy. >&2
      update_routing "${old_version}" 100

      echo Rolled back to ${old_version}
      exit 1
    fi

    echo Monitoring alarms...no alarms have triggered.
    sleep ${alarm_polling_interval}
    now=$(date +%s)
  done

  echo No alarms detected during canary period. Switching 100% of traffic to new
 version...
fi

update_routing "${new_version}" 100

echo Version ${new_version} is now receiving 100% of traffic.

if [[ (("${history_max}" -eq 0 ))]]; then
  echo Version History deletion is disabled. Remember to prune your history, the
 default limit is 1000 versions.
  echo Done!
  exit 0
fi

echo Keep the last ${history_max} versions. Deleting any versions older than that...

# the results are sorted in descending order of the version creation time
version_history=$($${aws} list-state-machine-versions --state-
machine-arn ${sm_arn} --max-results 1000 --query "join(\`\"\\n\"\`,
 stateMachineVersions[].stateMachineVersionArn)" --output text)

counter=0
```

```
while read line; do
  ((counter=${counter} + 1))

  if [[ (( ${counter} -gt ${history_max})) ]]; then
    echo Deleting old version ${line}
    ${aws} delete-state-machine-version --state-machine-version-arn ${line}
  fi
done <<< "${version_history}"

echo Done!
```

GunakanAWSSDK untuk menerapkan versi mesin status baru

Contoh script diaws-stepfunctions-examplesmenunjukkan bagaimana menggunakanAWSSDK untuk Python untuk secara bertahap menggeser lalu lintas dari versi sebelumnya ke versi baru dari mesin negara. Anda dapat menggunakan skrip contoh ini atau memperbaruinya sesuai dengan kebutuhan Anda.

Script menunjukkan strategi penyebaran berikut:

- Kenari- Pergeseran lalu lintas dalam dua bertahap.

  Pada kenaikan pertama, sebagian kecil lalu lintas, misalnya, 10 persen digeser ke versi baru. Dalam kenaikan kedua, sebelum interval waktu yang ditentukan dalam detik akan berakhir, lalu lintas yang tersisa digeser ke versi baru. Peralihan ke versi baru untuk lalu lintas yang tersisa hanya terjadi jika tidakCloudWatchalarm berangkat selama interval waktu yang ditentukan.
- Linear atau Bergulir- Menggeser lalu lintas ke versi baru dengan penambahan yang sama dengan jumlah detik yang sama di antara setiap kenaikan.

  Misalnya, jika Anda menentukan kenaikan persen sebagai**20**dengan`--interval`dari**600**detik, penyebaran ini meningkatkan lalu lintas sebesar 20 persen setiap 600 detik sampai versi baru menerima 100 persen lalu lintas.

  Penyebaran ini segera memutar kembali versi baru jika adaCloudWatchalarm berangkat.
- Semua Sekaligus atau Biru/Hijau- Pergeseran 100 persen lalu lintas ke versi baru segera. Penyebaran ini memonitor versi baru dan menggulungnya kembali secara otomatis ke versi sebelumnya jika adaCloudWatchalarm berangkat.

GunakanAWS CloudFormationuntuk menyebarkan versi mesin state baru

Berikut iniCloudFormationcontoh template menerbitkan dua versi dari mesin negara bernama*MyStateMachine*. Ini menciptakan sebuah alias bernama*PROD*, yang menunjuk ke kedua versi ini, dan kemudian menyebarkan versi2.

Dalam contoh ini, 10 persen lalu lintas digeser ke versi2setiap lima menit sampai versi ini menerima 100 persen dari lalu lintas. Contoh ini juga menunjukkan bagaimana Anda dapat mengaturCloudWatchalarm. Jika salah satu alarm yang Anda atur masuk keALARMnegara, penyebaran gagal dan gulungan kembali segera.

```
MyStateMachine:
  Type: AWS::StepFunctions::StateMachine
  Properties:
    Type: STANDARD
    StateMachineName: MyStateMachine
    RoleArn: arn:aws:iam::123456789012:role/myIamRole
    Definition:
      StartAt: PassState
      States:
        PassState:
          Type: Pass
          Result: Result
          End: true

MyStateMachineVersionA:
  Type: AWS::StepFunctions::StateMachineVersion
  Properties:
    Description: Version 1
    StateMachineArn: !Ref MyStateMachine

MyStateMachineVersionB:
  Type: AWS::StepFunctions::StateMachineVersion
  Properties:
    Description: Version 2
    StateMachineArn: !Ref MyStateMachine

PROD:
  Type: AWS::StepFunctions::StateMachineAlias
  Properties:
    Name: PROD
    Description: The PROD state machine alias taking production traffic.
    DeploymentPreference:
```

```
      StateMachineVersionArn: !Ref MyStateMachineVersionB
      Type: LINEAR
      Percentage: 10
      Interval: 5
      Alarms:
        # A list of alarms that you want to monitor. If any of these alarms trigger,
rollback the deployment immediately by pointing 100 percent of traffic to the previous
version.
        - !Ref CloudWatchAlarm1
        - !Ref CloudWatchAlarm2
```

Eksekusi di Step Functions

Eksekusi mesin status terjadi ketika sebuah mesin status AWS Step Functions berjalan dan melakukan tugasnya. Setiap mesin status Step Functions dapat memiliki beberapa eksekusi bersamaan, yang dapat Anda mulai dari konsol Step Functions, atau dengan menggunakan SDK AWS, tindakan API Step Functions, atau AWS Command Line Interface (AWS CLI). Eksekusi menerima input JSON dan menghasilkan output JSON. Anda dapat memulai eksekusi Step Functions dengan cara berikut:

- Panggil tindakan API StartExecution.
- Mulai eksekusi baru di konsol Step Functions.
- Gunakan Amazon EventBridge untuk memulai eksekusi sebagai respons terhadap suatu peristiwa.
- Gunakan Amazon EventBridge Scheduler untuk memulai eksekusi mesin negara sesuai jadwal.
- Mulai eksekusi dengan Amazon API Gateway.
- Mulai eksekusi alur kerja bersarang dari status Tugas.

Untuk informasi lebih lanjut tentang berbagai cara bekerja dengan Step Functions, lihat Opsi Pengembangan.

Mulai Eksekusi Alur Kerja dari Status Tugas.

AWS Step Functions dapat memulai eksekusi alur kerja secara langsung dari status `Task` mesin status. Hal ini mengizinkan Anda memecah alur kerja Anda menjadi mesin status yang lebih kecil, dan untuk memulai eksekusi dari mesin-mesin status lainnya. Dengan memulai eksekusi alur kerja baru ini, Anda dapat:

- Pisahkan alur kerja tingkat yang lebih tinggi dari tingkat yang lebih rendah, alur kerja khusus tugas.

- Hindari elemen berulang dengan memanggil mesin status terpisah beberapa kali.
- Buat pustaka alur kerja modular yang dapat digunakan kembali untuk pengembangan yang lebih cepat.
- Kurangi kompleksitas dan buat lebih mudah untuk mengedit dan memecahkan masalah mesin status.

Step Functions dapat memulai eksekusi alur kerja ini dengan memanggil API sendiri sebagai layanan terpadu. Cukup panggil `StartExecution`tindakan API dari status `Task` dan teruskan parameter yang diperlukan. Anda dapat memanggil API Step Functions menggunakan salah satu pola integrasi layanan.

> **Tip**
>
> Untuk menerapkan contoh alur kerja bertingkat ke AndaAkun AWS, lihat Modul 13 - Alur Kerja Ekspres Bertingkat.

Untuk memulai eksekusi baru dari mesin negara, gunakan `Task` status yang mirip dengan contoh berikut:

```
{
   "Type":"Task",
   "Resource":"arn:aws:states:::states:startExecution",
   "Parameters":{
      "StateMachineArn":"arn:aws:states:us-
east-1:123456789012:stateMachine:HelloWorld",
      "Input":{
         "Comment":"Hello world!"
      },
   },
   "Retry":[
      {
        "ErrorEquals":[
            "StepFunctions.ExecutionLimitExceeded"
        ]
      }
   ],
   "End":true
}
```

Status `Task` ini akan memulai eksekusi baru dari mesin status `HelloWorld`, dan akan meneruskan komentar JSON sebagai input.

> **Note**
>
> Kuota tindakan API `StartExecution` dapat membatasi jumlah eksekusi yang dapat Anda mulai. Gunakan opsi `Retry` pada `StepFunctions.ExecutionLimitExceeded` untuk memastikan eksekusi Anda dimulai. Lihat hal-hal berikut.
>
> - Kuota terkait throttling tindakan API
> - Penanganan kesalahan dalam Step Functions

Kaitkan Eksekusi Alur Kerja

Untuk mengaitkan eksekusi alur kerja yang dimulai dengan eksekusi yang memulainya, teruskan ID eksekusi dari objek konteks ke input eksekusi. Anda dapat mengakses ID dari objek konteks dari status `Task` Anda dalam eksekusi yang berjalan. Teruskan ID eksekusi dengan menambahkan `.$` ke nama parameter, dan mereferensikan ID dalam objek konteks dengan `$$.Execution.Id`.

```
"AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID.$": "$$.Execution.Id"
```

Anda dapat menggunakan parameter khusus bernama `AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID` ketika Anda memulai eksekusi. Jika disertakan, asosiasi ini menyediakan tautan di bagian Detail langkah dari konsol Step Functions. Jika tersedia, Anda dapat dengan mudah melacak eksekusi alur kerja Anda dari memulai eksekusi hingga eksekusi alur kerja yang dimulai. Menggunakan contoh sebelumnya, kaitkan ID eksekusi dengan eksekusi yang dimulai dari mesin status `HelloWorld`, sebagai berikut.

```
{
   "Type":"Task",
   "Resource":"arn:aws:states:::states:startExecution",
   "Parameters":{
      "StateMachineArn":"arn:aws:states:us-
east-1:123456789012:stateMachine:HelloWorld",
      "Input": {
        "Comment": "Hello world!",
        "AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID.$": "$$.Execution.Id"
       }
   },
```

```
   "End":true
}
```

Untuk informasi selengkapnya, lihat yang berikut:

- Bekerja dengan layanan yang lain
- Meneruskan parameter ke API layanan
- Mengakses Obyek Konteks
- AWS Step Functions

Menggunakan Amazon EventBridge Scheduler dengan AWS Step Functions

Amazon EventBridge Scheduler adalah penjadwal tanpa server yang memungkinkan Anda membuat, menjalankan, dan mengelola tugas dari satu layanan terpusat dan terkelola. Dengan EventBridge Scheduler, Anda dapat membuat jadwal menggunakan ekspresi cron dan rate untuk pola berulang, atau mengonfigurasi pemanggilan satu kali. Anda dapat mengatur jendela waktu fleksibel untuk pengiriman, menentukan batas coba lagi, dan mengatur waktu retensi maksimum untuk pemanggilan API yang gagal.

Misalnya, dengan EventBridge Scheduler, Anda dapat memulai eksekusi mesin status pada jadwal ketika peristiwa terkait keamanan terjadi atau untuk mengotomatiskan pekerjaan pemrosesan data.

Halaman ini menjelaskan cara menggunakan EventBridge Scheduler untuk memulai eksekusi mesin status Step Functions sesuai jadwal.

Topik

- Siapkan peran eksekusi
- Buat jadwal
- Sumber daya terkait

Siapkan peran eksekusi

Saat Anda membuat jadwal baru, EventBridge Scheduler harus memiliki izin untuk menjalankan operasi API targetnya atas nama Anda. Anda memberikan izin ini ke EventBridge Scheduler menggunakan peran eksekusi. Kebijakan izin yang Anda lampirkan ke peran eksekusi jadwal

menentukan izin yang diperlukan. Izin ini bergantung pada API target yang ingin Anda panggil EventBridge Scheduler.

Bila Anda menggunakan konsol EventBridge Scheduler untuk membuat jadwal, seperti dalam prosedur berikut, EventBridge Scheduler secara otomatis mengatur peran eksekusi berdasarkan target yang Anda pilih. Jika Anda ingin membuat jadwal menggunakan salah satu SDK EventBridge Penjadwal, atau AWS CLIAWS CloudFormation, Anda harus memiliki peran eksekusi yang ada yang memberikan izin yang diperlukan EventBridge Penjadwal untuk memanggil target. Untuk informasi selengkapnya tentang mengatur peran eksekusi secara manual untuk jadwal Anda, lihat Menyiapkan peran eksekusi di Panduan Pengguna EventBridge Penjadwal.

Buat jadwal

Untuk membuat jadwal dengan menggunakan konsol

1. Buka konsol Amazon EventBridge Scheduler di https://console.aws.amazon.com/scheduler/home.
2. Pada halaman Jadwal, pilih Buat jadwal.
3. Pada halaman Tentukan detail jadwal, di bagian Nama jadwal dan deskripsi, lakukan hal berikut:

   a. Untuk nama Jadwal, masukkan nama untuk jadwal Anda. Sebagai contoh, **`MyTestSchedule`**.

   b. (Opsional) Untuk Deskripsi, masukkan deskripsi untuk jadwal Anda. Sebagai contoh, **`My first schedule`**.

   c. Untuk grup Jadwal, pilih grup jadwal dari daftar dropdown. Jika Anda tidak memiliki grup, pilih default. Untuk membuat grup jadwal, pilih buat jadwal Anda sendiri.

   Anda menggunakan grup jadwal untuk menambahkan tag ke grup jadwal.

4. • Pilih opsi jadwal Anda.

Kejadian	Lakukan ini...	
Jadwal satu kali Jadwal satu kali memanggil target hanya sekali pada tanggal dan waktu yang Anda tentukan.	Untuk tanggal dan waktu, lakukan hal berikut: • Masukkan tanggal yang valid dalam `YYYY/MM/DD` format.	

Kejadian	Lakukan ini...	
	• Masukkan stempel waktu dalam format 24 jamhh:mm. • Untuk Timezone, pilih zona waktu.	

Kejadian	Lakukan ini...	
Jadwal berulang Jadwal berulang memanggil target pada tingkat yang Anda tentukan menggunakan cron ekspresi atau ekspresi tingkat.	a. Untuk jenis Jadwal, lakukan salah satu hal berikut: • Untuk menggunakan ekspresi cron untuk menentukan jadwal, pilih Jadwal berbasis Cron dan masukkan ekspresi cron. • Untuk menggunakan ekspresi laju untuk menentukan jadwal, pilih Jadwal berbasis tarif dan masukkan ekspresi laju. Untuk informasi selengkapnya tentang ekspresi cron dan rate, lihat Menjadwalkan jenis pada EventBridge Scheduler di Panduan Pengguna EventBridge Penjadwal Amazon. b. Untuk jendela waktu Fleksibel, pilih Nonaktif untuk mematikan opsi, atau pilih salah satu jendela waktu yang telah ditentukan sebelumnya. Misalnya, jika Anda memilih 15 menit dan	

Kejadian	Lakukan ini...	
	Anda menetapkan jadwal berulang untuk memanggil targetnya setiap jam sekali, jadwal berjalan dalam 15 menit setelah dimulainya setiap jam.	

5. (Opsional) Jika Anda memilih Jadwal berulang pada langkah sebelumnya, di bagian Jangka Waktu, lakukan hal berikut:

   a. Untuk Timezone, pilih zona waktu.

   b. Untuk Tanggal dan waktu mulai, masukkan tanggal yang valid dalam `YYYY/MM/DD` format, lalu tentukan stempel waktu dalam format 24 jam`hh:mm`.

   c. Untuk Tanggal dan waktu berakhir, masukkan tanggal yang valid dalam `YYYY/MM/DD` format, lalu tentukan stempel waktu dalam format 24 jam`hh:mm`.

6. Pilih Selanjutnya.

7. Pada halaman Select target, pilih operasi AWS API yang dipanggil EventBridge Scheduler:

   a. Pilih AWS Step Functions StartExecution.

   b. Di StartExecutionbagian ini, pilih mesin negara atau pilih Buat mesin negara baru.

      Saat ini, Anda tidak dapat menjalankan alur kerja Synchronous Express sesuai jadwal.

   c. Masukkan payload JSON untuk eksekusi. Bahkan jika mesin status Anda tidak memerlukan payload JSON, Anda masih harus menyertakan input dalam format JSON seperti yang ditunjukkan pada contoh berikut.

```
{
    "Comment": "sampleJSONData"
}
```

8. Pilih Selanjutnya.

9. Pada halaman Pengaturan, lakukan hal berikut:

   a. Untuk mengaktifkan jadwal, di bawah Status jadwal, alihkan Aktifkan jadwal.

b. Untuk mengonfigurasi kebijakan coba lagi untuk jadwal Anda, di bawah Kebijakan Coba lagi dan antrean surat mati (DLQ), lakukan hal berikut:

   - Beralih Coba Lagi.
   - Untuk usia maksimum acara, masukkan jam maksimum dan min yang harus disimpan oleh EventBridge Scheduler untuk menyimpan acara yang belum diproses.
   - Waktu maksimum adalah 24 jam.
   - Untuk percobaan ulang Maksimum, masukkan jumlah maksimum kali EventBridge Scheduler mencoba ulang jadwal jika target mengembalikan kesalahan.

     Nilai maksimumnya adalah 185 percobaan ulang.

   Dengan kebijakan coba lagi, jika jadwal gagal memanggil targetnya, EventBridge Scheduler menjalankan kembali jadwal. Jika dikonfigurasi, Anda harus mengatur waktu retensi maksimum dan mencoba ulang untuk jadwal.

c. Pilih tempat EventBridge Scheduler menyimpan acara yang tidak terkirim.

Opsi antrian surat mati (DLQ)	Lakukan ini...	
Jangan simpan	Pilih Tidak Ada.	
Simpan acara di tempat yang sama Akun AWS di mana Anda membuat jadwal	a. Pilih Pilih antrian Amazon SQS di saya Akun AWS sebagai DLQ. b. Pilih Nama Sumber Daya Amazon (ARN) dari antrian Amazon SQS.	

Opsi antrian surat mati (DLQ)	Lakukan ini...
Simpan acara di tempat yang berbeda Akun AWS dari tempat Anda membuat jadwal	a. Pilih Tentukan antrean Amazon SQS di lain Akun AWS sebagai DLQ. b. Masukkan Nama Sumber Daya Amazon (ARN) dari antrian Amazon SQS.

d. Untuk menggunakan kunci yang dikelola pelanggan untuk mengenkripsi input target Anda, di bawah Enkripsi, pilih Sesuaikan pengaturan enkripsi (lanjutan).

   Jika Anda memilih opsi ini, masukkan ARN kunci KMS yang ada atau pilih AWS KMS key Buat untuk menavigasi ke AWS KMS konsol. Untuk informasi selengkapnya tentang cara EventBridge Scheduler mengenkripsi data Anda saat istirahat, lihat Enkripsi saat istirahat di Panduan Pengguna EventBridge Penjadwal Amazon.

e. Agar EventBridge Scheduler membuat peran eksekusi baru untuk Anda, pilih Buat peran baru untuk jadwal ini. Kemudian, masukkan nama untuk nama Peran. Jika Anda memilih opsi ini, EventBridge Scheduler melampirkan izin yang diperlukan untuk target template Anda ke peran.

10. Pilih Selanjutnya.
11. Di halaman Tinjau dan buat jadwal, tinjau detail jadwal Anda. Di setiap bagian, pilih Edit untuk kembali ke langkah itu dan mengedit detailnya.
12. Pilih Buat jadwal.

    Anda dapat melihat daftar jadwal baru dan yang sudah ada di halaman Jadwal. Di bawah kolom Status, verifikasi bahwa jadwal baru Anda Diaktifkan.

Untuk mengonfirmasi bahwa EventBridge Scheduler memanggil mesin status, periksa log Amazon CloudWatch mesin status.

Sumber daya terkait

Untuk informasi selengkapnya tentang EventBridge Scheduler, lihat berikut ini:

- EventBridge Panduan Pengguna Scheduler
- EventBridge Referensi API Scheduler
- EventBridge Penetapan Harga Scheduler

Eksekusi Alur Kerja Standar dan Ekspres di konsol

Saat Anda membuat mesin status, Anda memilih Jenis Standar atau Ekspres. Jenis default untuk mesin negara adalah Standar. Mesin negara yang Tipe Standar disebut alur kerja Standar dan mesin status yang Tipe Ekspres disebut alur kerja Ekspres.

Untuk alur kerja Standar dan Ekspres, Anda menentukan mesin status Anda menggunakan. Amazon States Language Eksekusi mesin status Anda akan berperilaku berbeda tergantung pada Jenis yang Anda pilih.

> **Important**
>
> Jenis yang Anda pilih tidak dapat diubah setelah Anda membuat mesin status.

Untuk informasi selengkapnya tentang alur kerja Standar dan Ekspres, lihatAlur Kerja Standar vs Ekspres.

Riwayat eksekusi alur kerja Standar dicatat dalam Step Functions, sedangkan riwayat eksekusi alur kerja Express tidak dicatat dalam Step Functions. Untuk merekam riwayat eksekusi alur kerja Express, Anda harus mengonfigurasinya untuk mengirim log ke Amazon CloudWatch. Untuk informasi selengkapnya, lihat Logging menggunakanCloudWatchLog.

Setelah logging dikonfigurasi pada alur kerja Express, Anda dapat melihat eksekusinya di konsol Step Functions. Pengalaman konsol untuk melihat eksekusi alur kerja Express dan eksekusi alur kerja Standar serupa, kecuali untuk perbedaan dan batasan berikut.

> **Note**
>
> Karena data eksekusi untuk alur kerja Express ditampilkan menggunakan Wawasan CloudWatch Log, pemindaian log akan dikenakan biaya. Secara default, grup log Anda hanya mencantumkan eksekusi yang diselesaikan dalam tiga jam terakhir. Jika Anda menentukan rentang waktu yang lebih besar yang mencakup lebih banyak peristiwa eksekusi, biaya

Anda akan meningkat. Untuk informasi selengkapnya, lihat Log Terjual di bawah tab Log di halaman CloudWatch Harga danLogging menggunakanCloudWatchLog.

Daftar Isi

- Perbedaan pengalaman konsol
- Pertimbangan dan batasan untuk melihat eksekusi alur kerja Express

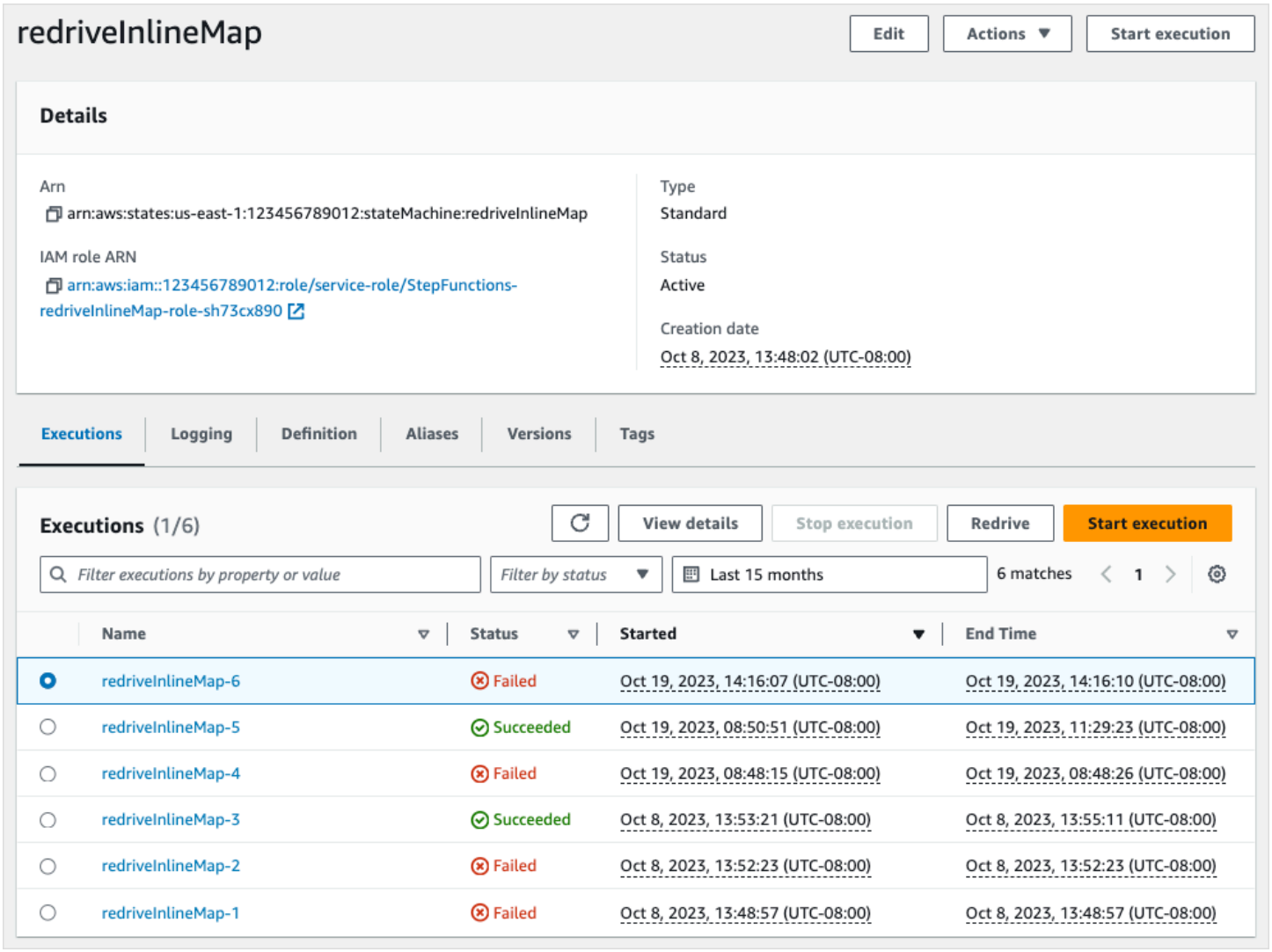
Perbedaan pengalaman konsol

Untuk semua alur kerja Standar dan Ekspres, Anda dapat melihat detail, seperti mesin status dan ARN peran IAM-nya, pada halaman detail mesin Status di konsol Step Functions.

Pada halaman detail mesin Status, Anda juga dapat melihat daftar riwayat eksekusi mesin status Anda di bawah tab Eksekusi. Gunakan kotak eksekusi Filter menurut properti atau nilai untuk mencari eksekusi, versi, atau alias tertentu dari mesin status yang dipilih. Gunakan dropdown Semua untuk memfilter riwayat eksekusi berdasarkan statusnya. Anda juga dapat memilih riwayat eksekusi dan memilih tombol Lihat detail untuk membuka halaman Detail eksekusi.

Alur kerja standar

Riwayat eksekusi untuk Alur Kerja Standar selalu tersedia untuk eksekusi yang diselesaikan dalam 90 hari terakhir.



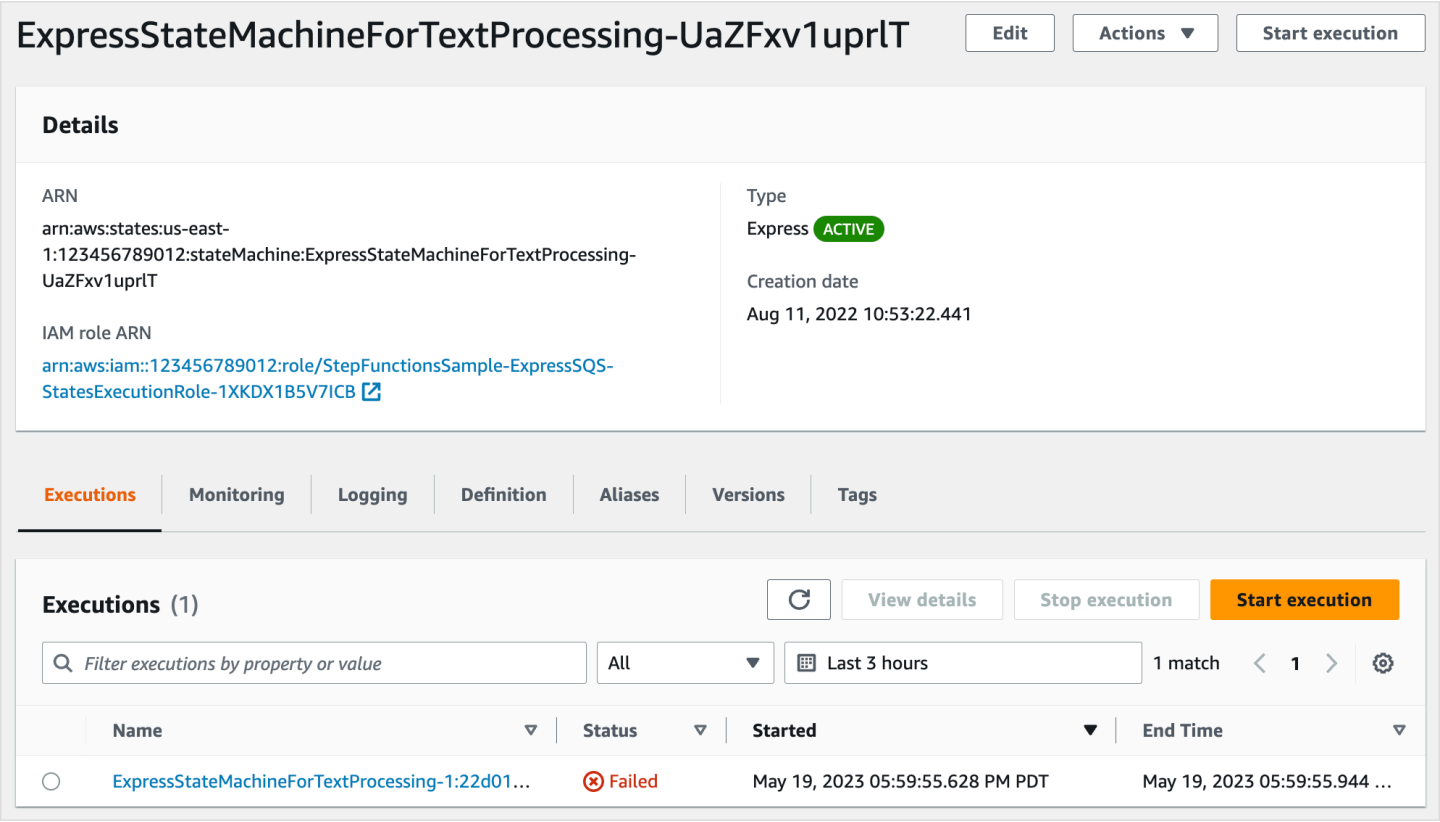
Alur kerja ekspres

Untuk menampilkan riwayat eksekusi untuk alur kerja Express, konsol Step Functions mengambil data log yang dikumpulkan melalui grup CloudWatch log Log.

Anda juga harus mengaktifkan pengalaman konsol baru untuk melihat eksekusi alur kerja Express. Untuk melakukan ini, pilih tombol Aktifkan yang ditampilkan di dalam spanduk pada tab Eksekusi. Setelah Anda memilih tombol ini, itu tidak akan muncul lagi.

> **Tip**
>
> Untuk beralih antara mengaktifkan atau menonaktifkan pengalaman konsol, gunakan tombol sakelar Aktifkan riwayat eksekusi ekspres.

Sejarah eksekusi yang diselesaikan dalam tiga jam terakhir tersedia secara default. Anda dapat menyesuaikan rentang waktu ini atau menentukan rentang khusus. Jika Anda menentukan rentang waktu yang lebih besar yang mencakup lebih banyak peristiwa eksekusi, biaya untuk memindai log akan meningkat. Untuk informasi selengkapnya, lihat Log Terjual di bawah tab Log di halaman CloudWatch Harga danLogging menggunakanCloudWatchLog.



Pertimbangan dan batasan untuk melihat eksekusi alur kerja Express

Saat melihat eksekusi alur kerja Express di konsol Step Functions, ingatlah pertimbangan dan batasan berikut.

- Ketersediaan detail eksekusi alur kerja Express bergantung pada Amazon Logs CloudWatch
- Rincian eksekusi alur kerja Partial Express tersedia jika level logging ERROR atau FATAL
- Definisi mesin status dari eksekusi yang lebih lama tidak dapat dilihat setelah diperbarui

Ketersediaan detail eksekusi alur kerja Express bergantung pada Amazon Logs CloudWatch

> **Note**
>
> Jika Anda tidak mengaktifkan pengalaman konsol baru untuk melihat eksekusi alur kerja Express, riwayat eksekusi dan detail eksekusi yang sesuai tidak akan tersedia di konsol Step Functions. Untuk mengaktifkan pengalaman konsol baru, pilih tombol Aktifkan yang ditampilkan di dalam spanduk pada tab Eksekusi.

Untuk alur kerja Express, riwayat eksekusi dan informasi eksekusi terperinci dikumpulkan melalui Wawasan CloudWatch Log. Informasi ini disimpan dalam grup CloudWatch log Log yang Anda tentukan saat Anda membuat mesin status. Riwayat eksekusi state machine ditampilkan di bawah tab Executions pada konsol Step Functions. Informasi terperinci tentang setiap eksekusi mesin negara ditampilkan pada halaman Detail eksekusi untuk eksekusi yang dipilih.

> **Warning**
>
> Jika Anda menghapus CloudWatch Log untuk alur kerja Ekspres, itu tidak akan terdaftar di bawah tab Eksekusi.

Kami menyarankan Anda menggunakan tingkat log default ALL untuk mencatat semua jenis peristiwa eksekusi. Anda dapat memperbarui tingkat log seperti yang diperlukan untuk mesin status yang ada saat Anda mengeditnya. Lihat informasi yang lebih lengkap di Logging menggunakanCloudWatchLog dan Tingkat Log.

Rincian eksekusi alur kerja Partial Express tersedia jika level logging ERROR atau FATAL

Secara default, tingkat logging untuk eksekusi alur kerja Express diatur ke ALL. Jika Anda mengubah level log, riwayat eksekusi dan detail eksekusi untuk eksekusi yang telah selesai tidak akan terpengaruh. Namun, semua eksekusi baru akan memancarkan log berdasarkan tingkat log yang diperbarui. Lihat informasi yang lebih lengkap di Logging menggunakanCloudWatchLog dan Tingkat Log.

Misalnya, jika Anda mengubah level log dari ALL menjadi ERROR atau FATAL, tab Eksekusi pada konsol Step Functions hanya mencantumkan eksekusi yang gagal. Di tab Tampilan acara, konsol hanya menampilkan detail peristiwa untuk langkah mesin status yang gagal.

Kami menyarankan Anda menggunakan tingkat log default ALL untuk mencatat semua jenis peristiwa eksekusi. Anda dapat memperbarui tingkat log seperti yang diperlukan untuk mesin status yang ada saat Anda mengedit mesin status.

Definisi mesin status dari eksekusi yang lebih lama tidak dapat dilihat setelah diperbarui

Definisi mesin status untuk eksekusi sebelumnya tidak disimpan untuk alur kerja Express. Jika Anda mengubah definisi mesin status, Anda hanya dapat melihat definisi mesin status untuk eksekusi menggunakan definisi terbaru.

Misalnya, jika Anda menghapus satu atau beberapa langkah dari definisi mesin status, Step Functions mendeteksi ketidakcocokan antara definisi dan peristiwa eksekusi sebelumnya. Karena definisi sebelumnya tidak disimpan untuk alur kerja Express, Step Functions tidak dapat menampilkan definisi mesin status untuk eksekusi yang dijalankan pada versi sebelumnya dari definisi mesin status. Akibatnya, tab input & output Eksekusi, Definisi, tampilan Grafik, dan tampilan Tabel tidak tersedia untuk eksekusi yang dijalankan pada versi definisi mesin status sebelumnya.

Melihat dan men-debug eksekusi di konsol Step Functions

Halaman Detail Eksekusi pada konsol Step Functions menyajikan informasi tentang eksekusi mesin status sebelumnya dan yang sedang berlangsung untuk Alur Kerja Standar dan Ekspres. Informasi ini ditampilkan dalam format dasbor. Misalnya, Anda dapat menemukan definisi Bahasa Negara Amazon dari mesin negara bagian, status eksekusi, ARN, dan jumlah total transisi status. Anda juga dapat melihat detail eksekusi untuk setiap negara bagian di mesin negara.

Daftar Isi

- Halaman Detail Eksekusi - Ikhtisar antarmuka
  - Ringkasan eksekusi
  - Pesan kesalahan
  - Mode tampilan
  - Detail langkah
  - Peristiwa
- Tutorial: Memeriksa eksekusi mesin status menggunakan konsol Step Functions
  - Langkah 1: Buat dan uji fungsi Lambda yang diperlukan
  - Langkah 2: Buat dan jalankan mesin negara
  - Langkah 3: Lihat detail eksekusi mesin negara

- Langkah 4: Jelajahi mode Tampilan yang berbeda

Halaman Detail Eksekusi - Ikhtisar antarmuka

Anda dapat menemukan detail untuk semua eksekusi mesin status yang sedang berlangsung dan sebelumnya untuk Alur Kerja Standar dan Ekspres di halaman Rincian Eksekusi. Jika Anda menentukan ID eksekusi saat memulai eksekusi, halaman ini diberi judul dengan ID eksekusi tersebut. Jika tidak, itu berjudul dengan ID eksekusi unik yang secara otomatis dihasilkan Step Functions untuk Anda.

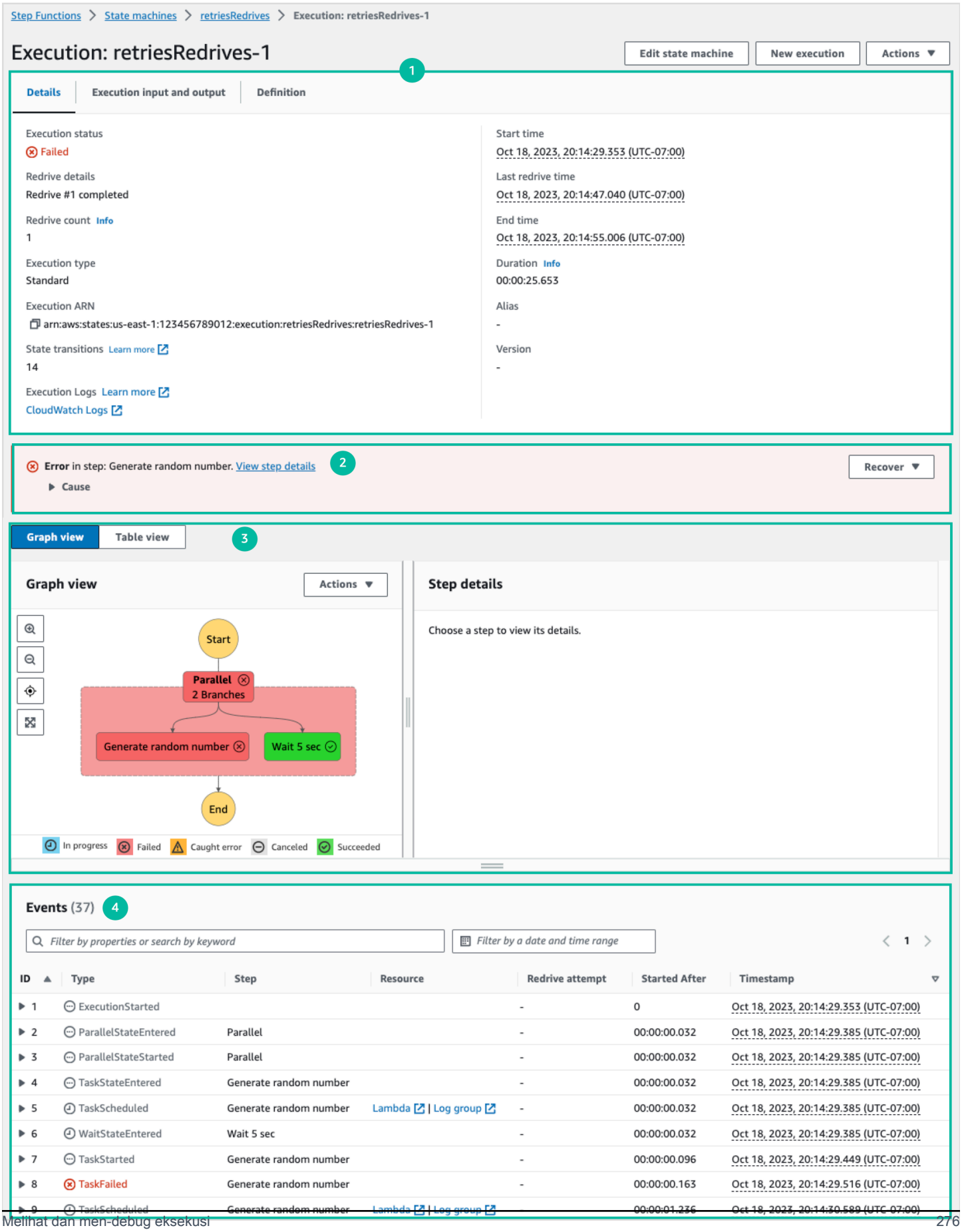
Selain metrik eksekusi, halaman Rincian Eksekusi menyediakan opsi berikut untuk mengelola mesin status Anda dan pelaksanaannya:

Tombol	Pilih tombol ini untuk:
Edit mesin negara	Edit definisi Bahasa Amazon States mesin negara bagian Anda.
Eksekusi baru	Mulai eksekusi baru dari mesin negara Anda.
Tindakan	Menyediakan opsi berikut untuk dipilih: • Hentikan eksekusi - Hentikan eksekusi yang sedang berlangsung. Opsi ini tidak tersedia untuk eksekusi yang diselesaikan. • Redrive— Redrive eksekusi Alur Kerja Standar yang tidak berhasil diselesai kan dalam 14 hari terakhir. Ini termasuk eksekusi yang gagal, dibatalkan, atau habis waktu. Untuk informasi selengkapnya, lihat Redrivingeksekusi. • Ekspor - Ekspor detail eksekusi dalam format JSON untuk membagikannya dengan orang lain atau melakukan analisis offline. • Kirim umpan balik - Bagikan umpan balik tentang antarmuka.

Melihat eksekusi dimulai dengan versi atau alias

Anda juga dapat melihat eksekusi yang dimulai dengan versi atau alias di konsol Step Functions. Untuk informasi selengkapnya, lihat Mencantumkan eksekusi untuk versi dan alias.

Halaman konsol Rincian Eksekusi berisi bagian-bagian berikut:


Step Functions > State machines > retriesRedrives > Execution: retriesRedrives-1
Execution: retriesRedrives-1
Edit state machine
New execution
Actions
1
Details
Execution input and output
Definition
Execution status
Failed
Redrive details
Redrive #1 completed
Redrive count Info
1
Execution type
Standard
Execution ARN
arn:aws:states:us-east-1:123456789012:execution:retriesRedrives:retriesRedrives-1
State transitions Learn more
14
Execution Logs Learn more
CloudWatch Logs
Start time
Oct 18, 2023, 20:14:29.353 (UTC-07:00)
Last redrive time
Oct 18, 2023, 20:14:47.040 (UTC-07:00)
End time
Oct 18, 2023, 20:14:55.006 (UTC-07:00)
Duration Info
00:00:25.653
Alias
-
Version
-
Error in step: Generate random number. View step details
2
Cause
Recover
Graph view
Table view
3
Graph view
Actions
Start
Parallel
2 Branches
Generate random number
Wait 5 sec
End
In progress
Failed
Caught error
Canceled
Succeeded
Step details
Choose a step to view its details.
Events (37)
4
Filter by properties or search by keyword
Filter by a date and time range
1
ID
Type
Step
Resource
Redrive attempt
Started After
Timestamp
1 ExecutionStarted - 0 Oct 18, 2023, 20:14:29.353 (UTC-07:00)
2 ParallelStateEntered Parallel - 00:00:00.032 Oct 18, 2023, 20:14:29.385 (UTC-07:00)
3 ParallelStateStarted Parallel - 00:00:00.032 Oct 18, 2023, 20:14:29.385 (UTC-07:00)
4 TaskStateEntered Generate random number - 00:00:00.032 Oct 18, 2023, 20:14:29.385 (UTC-07:00)
5 TaskScheduled Generate random number Lambda | Log group - 00:00:00.032 Oct 18, 2023, 20:14:29.385 (UTC-07:00)
6 WaitStateEntered Wait 5 sec - 00:00:00.032 Oct 18, 2023, 20:14:29.385 (UTC-07:00)
7 TaskStarted Generate random number - 00:00:00.096 Oct 18, 2023, 20:14:29.449 (UTC-07:00)
8 TaskFailed Generate random number - 00:00:00.163 Oct 18, 2023, 20:14:29.516 (UTC-07:00)
9 TaskScheduled Generate random number Lambda | Log group - 00:00:01.236 Oct 18, 2023, 20:14:30.589 (UTC-07:00)

1. Ringkasan eksekusi
2. Pesan kesalahan
3. Mode tampilan
4. Detail langkah
5. Peristiwa

Ringkasan eksekusi

Bagian ringkasan Eksekusi muncul di bagian atas halaman Rincian Eksekusi. Bagian ini memberi Anda gambaran umum tentang detail eksekusi alur kerja Anda. Informasi ini dibagi antara tiga tab berikut:

Detail

Menampilkan informasi, seperti status eksekusi, ARN, dan stempel waktu untuk waktu mulai dan berakhir eksekusi. Anda juga dapat melihat jumlah total transisi Status yang terjadi saat menjalankan eksekusi mesin status. Anda juga dapat melihat tautan untuk peta jejak X-Ray dan Log CloudWatch Eksekusi Amazon jika Anda mengaktifkan penelusuran atau log untuk mesin status Anda.

Jika eksekusi mesin status Anda dimulai oleh mesin status lain, Anda dapat melihat tautan untuk mesin status induk di tab ini.

Jika eksekusi mesin status Anda redriven, tab ini menampilkan informasi redrive terkait, misalnya Redrivehitungan.

Input dan output eksekusi

Menunjukkan input dan output eksekusi mesin negara side-by-side.

Definisi

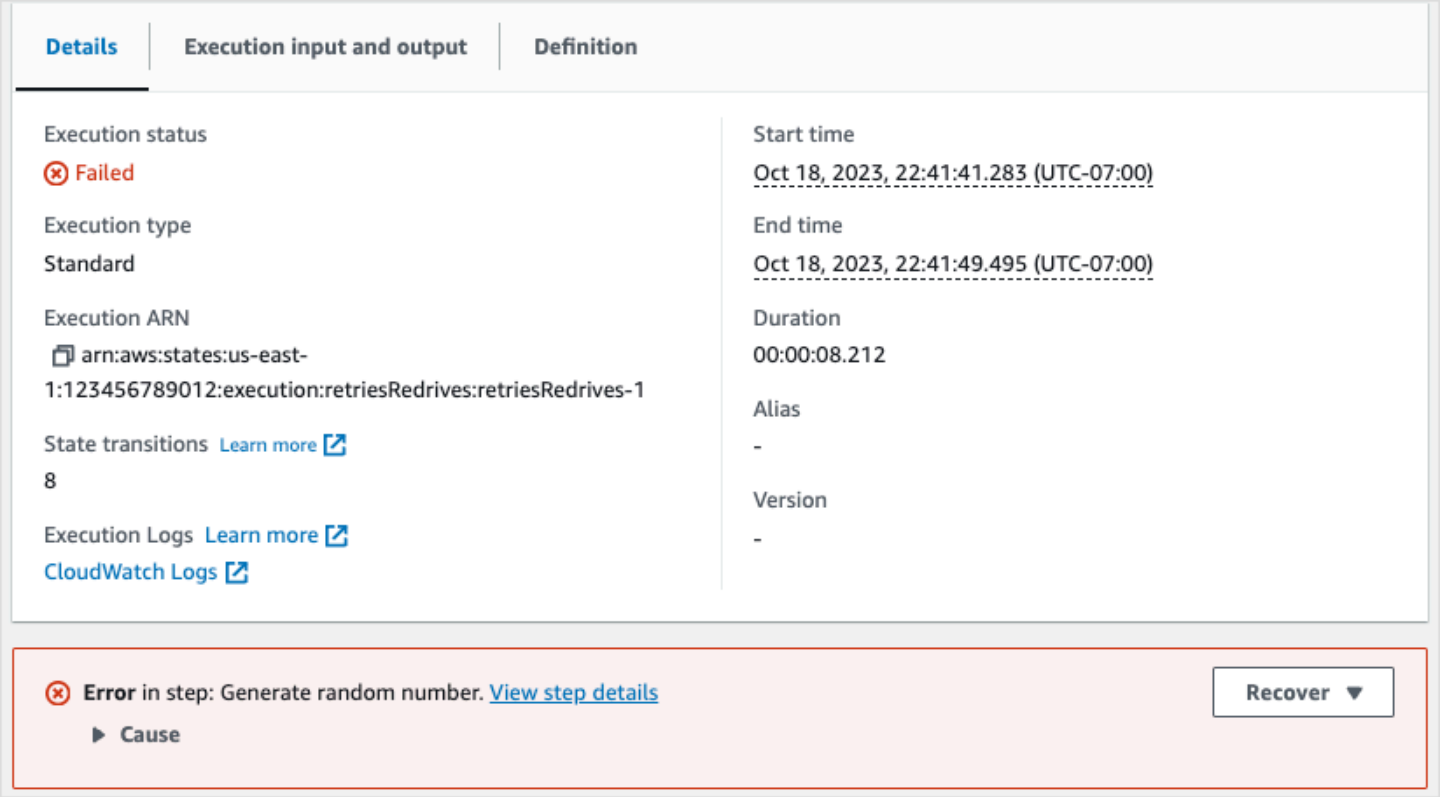
Menunjukkan definisi Bahasa Amazon States dari mesin negara bagian.

Pesan kesalahan

Jika eksekusi mesin status Anda gagal, halaman Rincian Eksekusi menampilkan pesan galat. Pilih Penyebab atau Lihat detail langkah dalam pesan kesalahan untuk melihat alasan kegagalan eksekusi atau langkah yang menyebabkan kesalahan.

Jika Anda memilih Lihat detail langkah, Step Functions menyoroti langkah yang menyebabkan kesalahan pada tab Detail langkah, tampilan Grafik, dan tampilan Tabel. Jika langkahnya adalah status Tugas, Peta, atau Paralel yang telah Anda tetapkan percobaan ulang, panel Detail langkah akan menampilkan tab Coba lagi untuk langkah tersebut. Selain itu, jika Anda memiliki redriven eksekusi, Anda dapat melihat percobaan ulang dan detail redrive eksekusi di tab Retries & redrives tab pada panel Detail langkah.

Dari tombol tarik-turun Pulihkan pada pesan kesalahan ini, Anda dapat mengeksekusi yang gagal atau memulai eksekusi baru. redrive Untuk informasi selengkapnya, lihat Redrivingeksekusi.



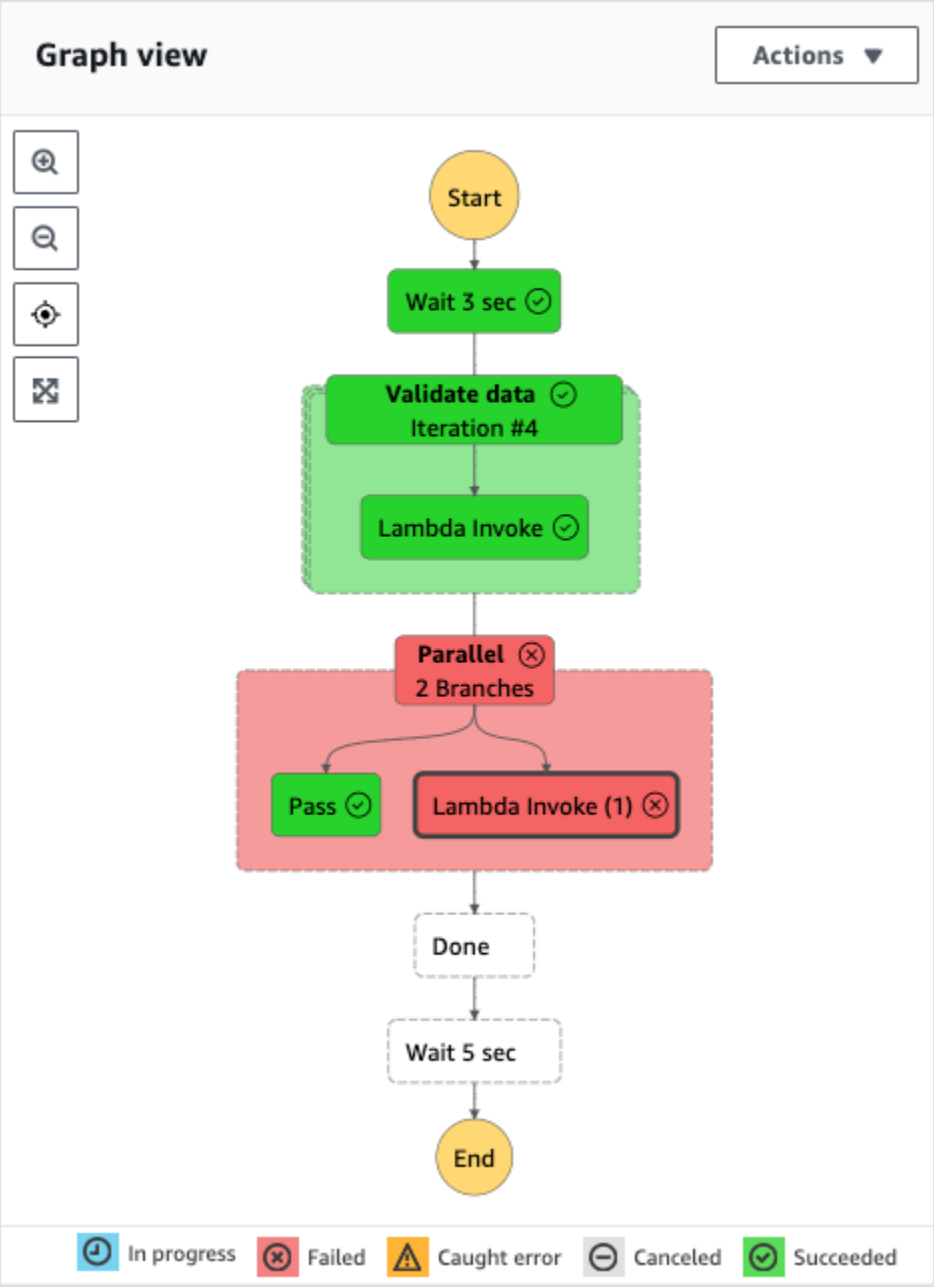
Mode tampilan

Bagian Mode Tampilan berisi dua visualisasi berbeda untuk mesin status Anda. Anda dapat memilih untuk melihat representasi grafis dari alur kerja, tabel yang menguraikan status dalam alur kerja Anda, atau daftar peristiwa yang terkait dengan eksekusi mesin status Anda:

> **Note**
>
> Pilih tab untuk melihat isinya.

Graph view

Mode tampilan Grafik menampilkan representasi grafis dari alur kerja Anda. Legenda disertakan di bagian bawah yang menunjukkan status eksekusi mesin negara. Ini juga berisi tombol yang memungkinkan Anda memperbesar, memperkecil, menyelaraskan alur kerja penuh, atau melihat alur kerja dalam mode layar penuh.



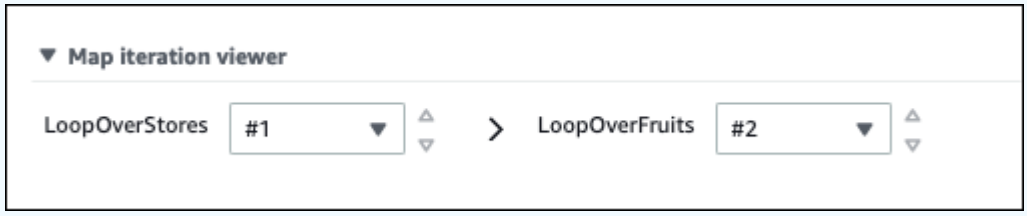
Dari tampilan ini, Anda dapat memilih langkah apa pun dalam alur kerja Anda untuk melihat detail tentang pelaksanaannya di komponen Detail langkah. Saat Anda memilih langkah dalam tampilan

Grafik, tampilan Tabel juga menunjukkan langkah itu. Hal ini juga berlaku secara terbalik. Jika Anda memilih langkah dari tampilan Tabel, tampilan Grafik menunjukkan langkah yang sama.

Jika mesin status Anda berisi `Map` status, `Parallel` status, atau keduanya, Anda dapat melihat namanya di alur kerja dalam tampilan Grafik. Selain itu, untuk `Map` status, tampilan Grafik memungkinkan Anda bergerak melintasi iterasi yang berbeda dari data eksekusi status Peta. Misalnya, jika status Peta Anda memiliki lima iterasi dan Anda ingin melihat data eksekusi untuk iterasi ketiga dan keempat, lakukan hal berikut:

1. Pilih status Peta yang data iterasinya ingin Anda lihat.
2. Dari Map iteration viewer, pilih #2 dari daftar dropdown untuk iterasi ketiga. Ini karena iterasi dihitung dari nol. Demikian juga, pilih #3 dari daftar dropdown untuk iterasi keempat dari status Peta.

   Atau, gunakan
   ▽
   kontrol
   △
   dan untuk berpindah di antara iterasi yang berbeda dari status Peta.

   > **Note**
   >
   > Jika mesin status Anda berisi `Map` status bersarang, daftar tarik-turun untuk iterasi `Map` status induk dan anak akan ditampilkan seperti yang ditunjukkan pada contoh berikut:
   >
   > 
   >

3. (Opsional) Jika satu atau beberapa iterasi status Peta gagal dijalankan, atau eksekusi dihentikan, Anda dapat melihat datanya dengan memilih nomor iterasi tersebut di bawah Gagal atau Dibatalkan dalam daftar tarik-turun.

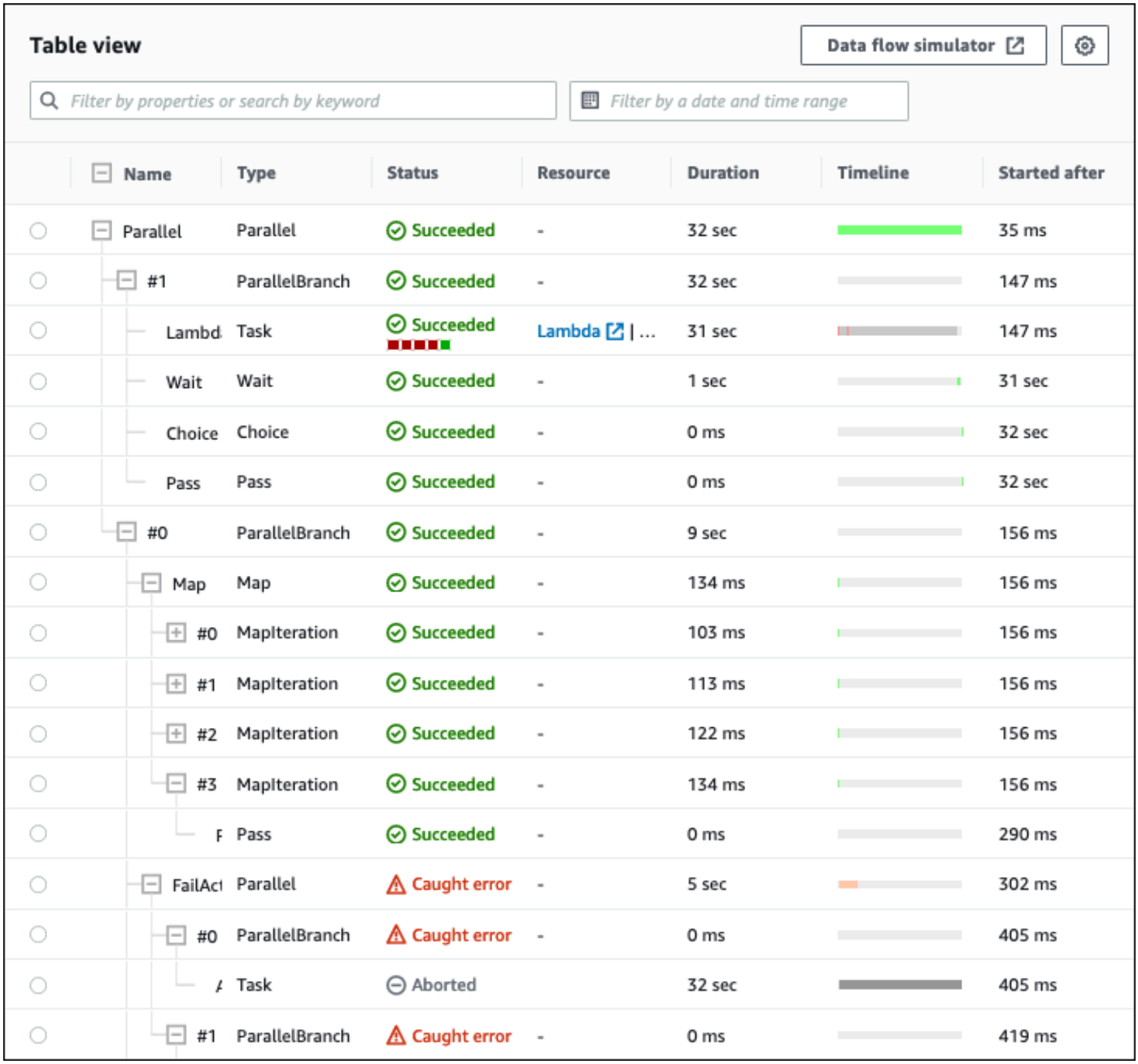
Terakhir, Anda dapat menggunakan tombol Ekspor dan Tata Letak untuk mengekspor grafik alur kerja sebagai gambar SVG atau PNG. Anda juga dapat beralih antara tampilan horizontal dan vertikal alur kerja Anda.

Table view

Mode tampilan Tabel menampilkan representasi tabel status dalam alur kerja Anda. Dalam mode Tampilan ini, Anda dapat melihat detail setiap status yang dieksekusi dalam alur kerja Anda, termasuk namanya, nama sumber daya apa pun yang digunakan (seperti AWS Lambda fungsi), dan jika status berhasil dijalankan.

Dari tampilan ini, Anda dapat memilih status apa pun dalam alur kerja Anda untuk melihat detail tentang pelaksanaannya di komponen Detail langkah. Saat Anda memilih langkah dalam tampilan Tabel, tampilan Grafik juga menunjukkan langkah itu. Hal ini juga berlaku secara terbalik. Jika Anda memilih langkah dari tampilan Grafik, tampilan Tabel menunjukkan langkah yang sama.

Anda juga dapat membatasi jumlah data yang ditampilkan dalam mode tampilan Tabel dengan menerapkan filter ke tampilan. Anda dapat membuat filter untuk properti tertentu, seperti Status atau Redriveupaya. Untuk informasi selengkapnya, lihat Tutorial: Memeriksa eksekusi mesin status menggunakan konsol Step Functions.

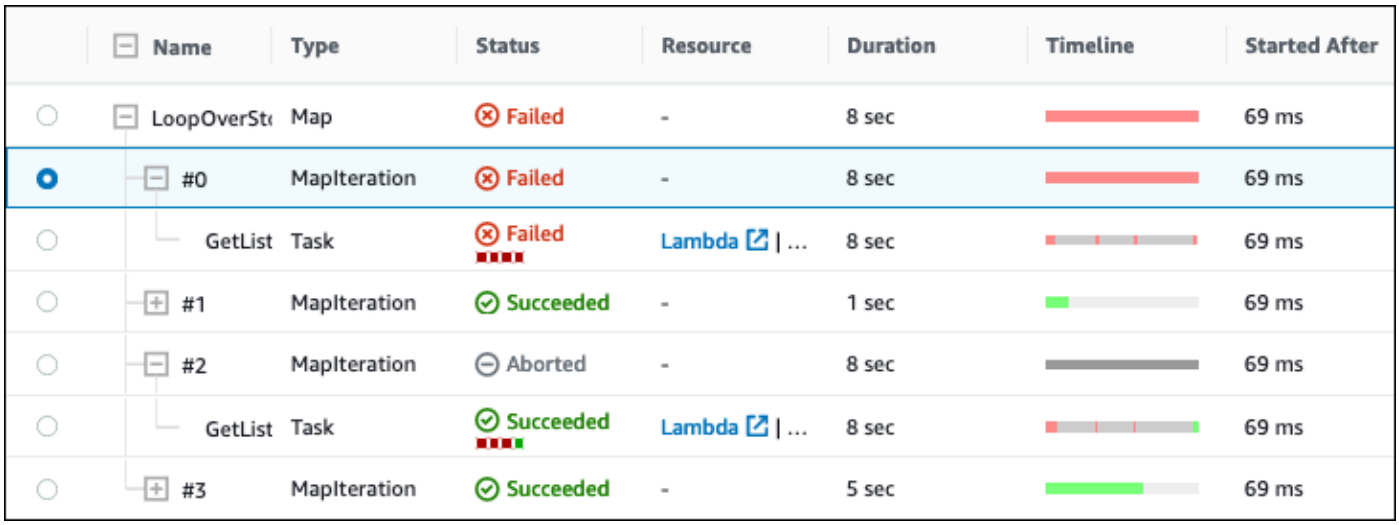


Secara default, mode ini menampilkan kolom Nama, Jenis, Status, Sumber Daya, dan Dimulai Setelah. Anda dapat mengonfigurasi kolom yang ingin Anda lihat menggunakan kotak dialog Preferensi. Pilihan yang Anda buat pada kotak dialog ini bertahan untuk eksekusi mesin status masa depan hingga diubah lagi.

Jika Anda menambahkan kolom Timeline, durasi eksekusi setiap status ditampilkan sehubungan dengan runtime untuk seluruh eksekusi. Ini ditampilkan sebagai garis waktu linier berkode warna. Ini dapat membantu Anda mengidentifikasi masalah terkait kinerja dengan eksekusi status

tertentu. Segmen kode warna untuk setiap status pada timeline membantu Anda mengidentifikasi status eksekusi status, seperti sedang berlangsung, gagal, atau dibatalkan.

Misalnya, jika Anda telah menetapkan percobaan ulang eksekusi untuk status di mesin status Anda, percobaan ulang ini ditampilkan di garis waktu. Segmen merah mewakili `Retry` upaya yang gagal, sedangkan segmen abu-abu muda mewakili `BackoffRate` antara setiap `Retry` upaya.



Jika mesin status Anda berisi `Map` status, `Parallel` status, atau keduanya, Anda dapat melihat namanya dalam alur kerja dalam tampilan Tabel. Untuk `Map` dan `Parallel` menyatakan, mode tampilan Tabel menampilkan data eksekusi untuk iterasi dan cabang paralelnya sebagai node di dalam tampilan pohon. Anda dapat memilih setiap node di negara bagian ini untuk melihat detail masing-masing di bagian Detail langkah. Misalnya, Anda dapat meninjau data untuk iterasi status Peta tertentu yang menyebabkan status gagal. Perluas node untuk status Peta, lalu lihat status untuk setiap iterasi di kolom Status.

Detail langkah

Bagian Detail langkah terbuka di sebelah kanan saat Anda memilih status dalam tampilan Grafik atau tampilan Tabel. Bagian ini berisi tab berikut, yang memberi Anda informasi mendalam tentang status yang dipilih:

Masukan

Menampilkan rincian masukan dari status yang dipilih. Jika ada kesalahan dalam input, itu ditunjukkan dengan header

⚠

pada tab. Selain itu, Anda dapat melihat alasan kesalahan di tab ini.

Anda juga dapat memilih tombol sakelar Tampilan lanjutan untuk melihat jalur transfer data input saat data melewati status yang dipilih. Ini memungkinkan Anda mengidentifikasi bagaimana input Anda diproses sebagai satu atau beberapa bidang, seperti`InputPath`,, `ParametersResultSelector`,`OutputPath`,`ResultPath`, dan, diterapkan ke data.

Keluaran

Menampilkan output dari status yang dipilih. Jika ada kesalahan dalam output, itu ditunjukkan dengan header

⚠

pada tab. Selain itu, Anda dapat melihat alasan kesalahan di tab ini.

Anda juga dapat memilih tombol sakelar Tampilan lanjutan untuk melihat jalur transfer data keluaran saat data melewati status yang dipilih. Ini memungkinkan Anda mengidentifikasi bagaimana input Anda diproses sebagai satu atau beberapa bidang, seperti`InputPath`,, `ParametersResultSelector`,`OutputPath`,`ResultPath`, dan, diterapkan ke data.

Detail

Menampilkan informasi, seperti jenis status, status eksekusi, dan durasi eksekusi.

Untuk `Task` status yang menggunakan sumber daya, seperti AWS Lambda, tab ini menyediakan tautan ke halaman definisi sumber daya dan halaman CloudWatch log Amazon untuk pemanggilan sumber daya. Ini juga menunjukkan nilai, jika ditentukan, untuk `Task` negara bagian `TimeoutSeconds` dan `HeartbeatSeconds` bidang.

Untuk `Map` status, tab ini menunjukkan informasi mengenai jumlah total iterasi `Map` status. Iterasi dikategorikan sebagai Gagal, Dibatalkan, Berhasil, atau. InProgress

Definisi

Menampilkan definisi Bahasa Negara Amazon yang sesuai dengan status yang dipilih.

Coba lagi

> **Note**
>
> Tab ini hanya muncul jika Anda telah menentukan `Retry` bidang di mesin status `Task` atau `Parallel` status Anda.

Menunjukkan percobaan ulang awal dan berikutnya untuk status yang dipilih dalam upaya eksekusi aslinya. Untuk upaya awal dan semua upaya gagal berikutnya, pilih di

▶

sebelah Jenis untuk melihat Alasan kegagalan yang muncul di kotak tarik-turun. Jika upaya coba lagi berhasil, Anda dapat melihat Output yang muncul di kotak dropdown.

Jika redriven Anda memiliki eksekusi, header tab ini menampilkan nama Retries & redrives dan menampilkan detail percobaan ulang untuk masing-masing. redrive

Peristiwa

Menampilkan daftar peristiwa yang difilter terkait dengan status yang dipilih dalam eksekusi. Informasi yang Anda lihat di tab ini adalah bagian dari riwayat peristiwa eksekusi lengkap yang Anda lihat di tabel Peristiwa.

Peristiwa

Tabel Acara menampilkan riwayat lengkap untuk eksekusi yang dipilih sebagai daftar peristiwa yang mencakup beberapa halaman. Setiap halaman berisi hingga 25 acara. Bagian ini juga menampilkan total jumlah acara, yang dapat membantu Anda menentukan apakah Anda melebihi jumlah riwayat acara maksimum 25.000 peristiwa.

Events (109)

Filter by properties or search by keyword | Filter by a date and time range | 1

ID	Type	Step	Resource	Redrive attempt	Started After	Timestamp
▶ 95	TaskStateAborted			#2	02:37:37.672	Oct 19, 2023, 11:28:28.958 (UTC-07:00)
▶ 96	ParallelStateFailed	Parallel		#2	02:37:37.672	Oct 19, 2023, 11:28:28.958 (UTC-07:00)
▶ 97	ExecutionFailed			#2	02:37:37.713	Oct 19, 2023, 11:28:28.999 (UTC-07:00)
▶ 98	ExecutionRedriven			#3	02:38:24.882	Oct 19, 2023, 11:29:16.168 (UTC-07:00)
▶ 99	TaskScheduled	Lambda Invoke (1)	Lambda | Log group	#3	02:38:24.904	Oct 19, 2023, 11:29:16.190 (UTC-07:00)
▶ 100	TaskStarted	Lambda Invoke (1)		#3	02:38:24.985	Oct 19, 2023, 11:29:16.271 (UTC-07:00)
▶ 101	TaskSucceeded	Lambda Invoke (1)		#3	02:38:27.260	Oct 19, 2023, 11:29:18.546 (UTC-07:00)
▶ 102	TaskStateExited	Lambda Invoke (1)		#3	02:38:27.282	Oct 19, 2023, 11:29:18.568 (UTC-07:00)
▶ 103	ParallelStateSucceeded	Parallel		#3	02:38:27.282	Oct 19, 2023, 11:29:18.568 (UTC-07:00)
▶ 104	ParallelStateExited	Parallel		#3	02:38:27.282	Oct 19, 2023, 11:29:18.568 (UTC-07:00)
▶ 105	PassStateEntered	Done		#3	02:38:27.282	Oct 19, 2023, 11:29:18.568 (UTC-07:00)
▶ 106	PassStateExited	Done		#3	02:38:27.282	Oct 19, 2023, 11:29:18.568 (UTC-07:00)
▶ 107	WaitStateEntered	Wait 5 sec		#3	02:38:27.282	Oct 19, 2023, 11:29:18.568 (UTC-07:00)
▶ 108	WaitStateExited	Wait 5 sec		#3	02:38:32.345	Oct 19, 2023, 11:29:23.631 (UTC-07:00)
▶ 109	ExecutionSucceeded			#3	02:38:32.394	Oct 19, 2023, 11:29:23.680 (UTC-07:00)

Secara default, hasil dalam tabel Peristiwa ditampilkan dalam urutan menaik berdasarkan Stempel Waktu peristiwa. Anda dapat mengubah pengurutan riwayat peristiwa eksekusi menjadi urutan menurun dengan mengklik header kolom Timestamp.

Dalam tabel Peristiwa, setiap peristiwa diberi kode warna untuk menunjukkan status eksekusinya. Misalnya, peristiwa yang gagal muncul dalam warna merah. Untuk melihat detail tambahan tentang suatu peristiwa, pilih di

sebelah ID acara. Setelah terbuka, detail acara menunjukkan input, output, dan pemanggilan sumber daya untuk acara tersebut.

Selain itu, di tabel Acara, Anda dapat menerapkan filter untuk membatasi hasil riwayat peristiwa eksekusi yang ditampilkan. Anda dapat memilih properti seperti ID, atau Redrivemencoba. Lihat informasi yang lebih lengkap di Tutorial: Memeriksa eksekusi mesin status menggunakan konsol Step Functions.

Tutorial: Memeriksa eksekusi mesin status menggunakan konsol Step Functions

Dalam tutorial ini, Anda akan belajar bagaimana memeriksa informasi eksekusi yang ditampilkan pada halaman Rincian Eksekusi dan melihat alasan eksekusi gagal. Kemudian, Anda akan belajar cara mengakses iterasi yang berbeda dari eksekusi `Map` status. Terakhir, Anda akan belajar cara mengonfigurasi kolom pada tampilan Tabel dan menerapkan filter yang sesuai untuk hanya melihat informasi yang menarik bagi Anda.

Dalam tutorial ini, Anda membuat mesin status tipe Standar, yang memperoleh harga satu set buah. Untuk melakukan ini, mesin negara menggunakan tiga AWS Lambda fungsi yang mengembalikan daftar acak empat buah, harga setiap buah, dan biaya rata-rata buah. Fungsi Lambda dirancang untuk menimbulkan kesalahan jika harga buah kurang dari atau sama dengan nilai ambang batas.

> **Note**
>
> Meskipun prosedur berikut berisi petunjuk tentang cara memeriksa detail eksekusi alur kerja Standar, Anda juga dapat memeriksa detail untuk eksekusi alur kerja Express. Untuk informasi tentang perbedaan antara detail eksekusi untuk tipe alur kerja Standar dan Ekspres, lihatEksekusi Alur Kerja Standar dan Ekspres di konsol.

Daftar Isi

- Langkah 1: Buat dan uji fungsi Lambda yang diperlukan

- Langkah 2: Buat dan jalankan mesin negara
- Langkah 3: Lihat detail eksekusi mesin negara
- Langkah 4: Jelajahi mode Tampilan yang berbeda

Langkah 1: Buat dan uji fungsi Lambda yang diperlukan

1. Buka konsol Lambda dan kemudian lakukan langkah 1 hingga 4 di bagian tersebut. Langkah 1: Membuat fungsi Lambda Pastikan untuk memberi nama fungsi Lambda. **`GetListOfFruits`**
2. Setelah Anda membuat fungsi Lambda, salin Nama Sumber Daya Amazon (ARN) fungsi yang ditampilkan di sudut kanan atas halaman. Untuk menyalin ARN, klik tombol.

   Berikut ini adalah contoh ARN, di mana *`function-name`*adalah nama fungsi Lambda (dalam hal ini,): `GetListOfFruits`

   ```
   arn:aws:lambda:us-east-1:123456789012:function:function-name
   ```

3. Salin kode berikut untuk fungsi Lambda ke area sumber Kode halaman. GetListOfFruits

   ```
   function getRandomSubarray(arr, size) {
       var shuffled = arr.slice(0), i = arr.length, temp, index;
       while (i--) {
           index = Math.floor((i + 1) * Math.random());
           temp = shuffled[index];
           shuffled[index] = shuffled[i];
           shuffled[i] = temp;
       }
       return shuffled.slice(0, size);
   }

   exports.handler = async function(event, context) {

       const fruits = ['Abiu','Açaí','Acerola','Ackee','African
    cucumber','Apple','Apricot','Avocado','Banana','Bilberry','Blackberry','Blackcurrant','Jos


        const errorChance = 45;

       const waitTime = Math.floor( 100 * Math.random() );

       await new Promise( r => setTimeout(() => r(), waitTime));
   ```

```
    const num = Math.floor( 100 * Math.random() );
    // const num = 51;
     if (num <= errorChance) {
         throw(new Error('Error'));
     }

    return getRandomSubarray(fruits, 4);
};
```

4. Pilih Deploy, lalu pilih Test, untuk menyebarkan perubahan dan melihat output dari fungsi Lambda Anda.
5. Buat dua fungsi Lambda tambahan, bernama **`GetFruitPrice`** dan **`CalculateAverage`** masing-masing, dengan langkah-langkah berikut:

   a. Salin kode berikut ke area sumber Kode dari fungsi GetFruitPriceLambda:

```
exports.handler = async function(event, context) {

    const errorChance = 0;
    const waitTime = Math.floor( 100 * Math.random() );

    await new Promise( r => setTimeout(() => r(), waitTime));

    const num = Math.floor( 100 * Math.random() );
    if (num <= errorChance) {
        throw(new Error('Error'));
    }

    return Math.floor(Math.random()*100)/10;
};
```

   b. Salin kode berikut ke area sumber Kode dari fungsi CalculateAverageLambda:

```
function getRandomSubarray(arr, size) {
    var shuffled = arr.slice(0), i = arr.length, temp, index;
    while (i--) {
        index = Math.floor((i + 1) * Math.random());
        temp = shuffled[index];
        shuffled[index] = shuffled[i];
        shuffled[i] = temp;
    }
    return shuffled.slice(0, size);
```

```
}

const average = arr => arr.reduce( ( p, c ) => p + c, 0 ) / arr.length;

exports.handler = async function(event, context) {
        const errors = [
        "Error getting data from DynamoDB",
        "Error connecting to DynamoDB",
        "Network error",
        "MemoryError - Low memory"
        ]

    const errorChance = 0;

    const waitTime = Math.floor( 100 * Math.random() );

    await new Promise( r => setTimeout(() => r(), waitTime));

    const num = Math.floor( 100 * Math.random() );
    if (num <= errorChance) {
        throw(new Error(getRandomSubarray(errors, 1)[0]));
    }

    return average(event);
};
```

c. Pastikan untuk menyalin ARN dari dua fungsi Lambda ini, lalu Deploy dan Test mereka.

Langkah 2: Buat dan jalankan mesin negara

Gunakan konsol Step Functions untuk membuat mesin status yang memanggil fungsi Lambda yang Anda buat di Langkah 1. Dalam mesin negara ini, tiga `Map` negara didefinisikan. Masing-masing `Map` status ini berisi `Task` status yang memanggil salah satu fungsi Lambda Anda. Selain itu, `Retry` bidang didefinisikan di setiap `Task` negara bagian dengan sejumlah upaya coba lagi yang ditentukan untuk setiap status. Jika `Task` status mengalami kesalahan runtime, status akan dieksekusi lagi hingga jumlah percobaan ulang yang ditentukan untuk itu. `Task`

1. Buka konsol Step Functions dan pilih Tulis alur kerja Anda dalam kode.

> **Important**
>
> Pastikan mesin status Anda berada di bawah AWS akun dan Wilayah yang sama dengan fungsi Lambda yang Anda buat sebelumnya.

2. Untuk Type, pertahankan pilihan standar Standar.
3. Salin definisi Bahasa Amazon States berikut dan tempel di bawah Definisi. Pastikan untuk mengganti ARN yang ditampilkan dengan fungsi Lambda yang sebelumnya Anda buat.

```
{
    "StartAt": "LoopOverStores",
    "States": {
        "LoopOverStores": {
            "Type": "Map",
            "Iterator": {
                "StartAt": "GetListOfFruits",
                "States": {
                    "GetListOfFruits": {
                        "Type": "Task",
                        "Resource": "arn:aws:states:::lambda:invoke",
                        "OutputPath": "$.Payload",
                        "Parameters": {
                            "FunctionName": "arn:aws:lambda:us-
east-1:123456789012:function:GetListofFruits:$LATEST",
                            "Payload": {
                                "storeName.$": "$"
                            }
                        },
                        "Retry": [
                            {
                                "ErrorEquals": [
                                    "States.ALL"
                                ],
                                "IntervalSeconds": 2,
                                "MaxAttempts": 1,
                                "BackoffRate": 1.3
                            }
                        ],
                        "Next": "LoopOverFruits"
                    },
                    "LoopOverFruits": {
```

```
                        "Type": "Map",
                        "Iterator": {
                            "StartAt": "GetFruitPrice",
                            "States": {
                                "GetFruitPrice": {
                                    "Type": "Task",
                                    "Resource": "arn:aws:states:::lambda:invoke",
                                    "OutputPath": "$.Payload",
                                    "Parameters": {
                                        "FunctionName": "arn:aws:lambda:us-
east-1:123456789012:function:GetFruitPrice:$LATEST",
                                        "Payload": {
                                            "fruitName.$": "$"
                                        }
                                    },
                                    "Retry": [
                                        {
                                            "ErrorEquals": [
                                                "States.ALL"
                                            ],
                                            "IntervalSeconds": 2,
                                            "MaxAttempts": 3,
                                            "BackoffRate": 1.3
                                        }
                                    ],
                                    "End": true
                                }
                            }
                        },
                        "ItemsPath": "$",
                        "End": true
                    }
                }
            },
            "ItemsPath": "$.stores",
            "Next": "LoopOverStoreFruitsPrice",
            "ResultPath": "$.storesFruitsPrice"
        },
        "LoopOverStoreFruitsPrice": {
            "Type": "Map",
            "End": true,
            "Iterator": {
                "StartAt": "CalculateAverage",
                "States": {
```

```
                    "CalculateAverage": {
                        "Type": "Task",
                        "Resource": "arn:aws:states:::lambda:invoke",
                        "OutputPath": "$.Payload",
                        "Parameters": {
                            "FunctionName": "arn:aws:lambda:us-
east-1:123456789012:function:Calculate-average:$LATEST",
                            "Payload.$": "$"
                        },
                        "Retry": [
                            {
                                "ErrorEquals": [
                                    "States.ALL"
                                ],
                                "IntervalSeconds": 2,
                                "MaxAttempts": 2,
                                "BackoffRate": 1.3
                            }
                        ],
                        "End": true
                    }
                }
            },
            "ItemsPath": "$.storesFruitsPrice",
            "ResultPath": "$.storesPriceAverage",
            "MaxConcurrency": 1
        }
    }
}
```

4. Masukkan nama untuk mesin negara Anda. Simpan pilihan default untuk opsi lain di halaman ini dan pilih Buat mesin status.

5. Buka halaman berjudul dengan nama mesin negara Anda. Lakukan langkah 1 hingga 4 di Langkah 4: Jalankan mesin negara bagian, tetapi gunakan data berikut sebagai input eksekusi:

```
{
    "stores": [
      "Store A",
      "Store B",
      "Store C",
      "Store D"
    ]
```

```
}
```

Langkah 3: Lihat detail eksekusi mesin negara

Pada halaman berjudul dengan ID eksekusi Anda, Anda dapat meninjau hasil eksekusi Anda dan men-debug kesalahan apa pun.

1. (Opsional) Pilih dari tab yang ditampilkan di halaman Rincian Eksekusi untuk melihat informasi yang ada di masing-masing tab tersebut. Misalnya, untuk melihat input mesin status dan output eksekusinya, pilih Input & output eksekusi pada bagian Ringkasan eksekusi.
2. Jika eksekusi mesin status Anda gagal, pilih Penyebab atau Tampilkan detail langkah pada pesan kesalahan. Detail tentang kesalahan ditampilkan di bagian Detail langkah. Perhatikan bahwa langkah yang menyebabkan kesalahan, yang merupakan `Task` status bernama GetListofFruits, disorot dalam tampilan Grafik dan tampilan Tabel.

> **Note**
>
> Karena GetListofFruitslangkah didefinisikan di dalam `Map` status, dan langkah gagal dijalankan dengan sukses, langkah Status status ditampilkan sebagai Gagal. `Map`

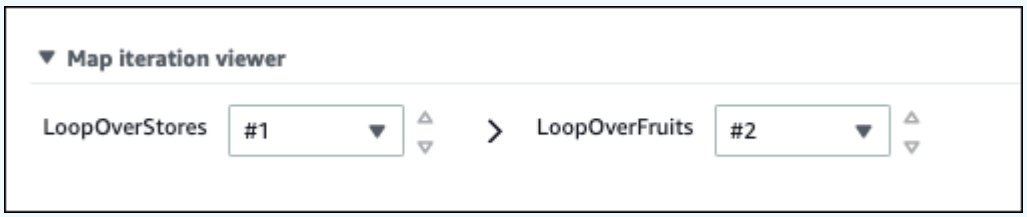
Langkah 4: Jelajahi mode Tampilan yang berbeda

Anda dapat memilih mode yang disukai untuk melihat alur kerja mesin status atau riwayat peristiwa eksekusi. Beberapa tugas yang dapat Anda lakukan dalam mode Tampilan ini adalah sebagai berikut:

Tampilan grafik - Beralih di antara iterasi **`Map`** status yang berbeda

Jika status Peta Anda memiliki lima iterasi dan Anda ingin melihat detail eksekusi untuk iterasi ketiga dan keempat, lakukan hal berikut:

1. Pilih `Map` status yang ingin Anda lihat data iterasi.
2. Dari Penampil iterasi Peta, pilih iterasi yang ingin Anda lihat. Iterasi dihitung dari nol. Untuk memilih iterasi ketiga dari lima, pilih #2 dari daftar tarik-turun di sebelah nama status Peta.

> **Note**
>
> Jika mesin status Anda berisi `Map` status bersarang, Step Functions menampilkan iterasi `Map` status induk dan anak sebagai dua daftar dropdown terpisah:
>
> 
>

3. (Opsional) Jika satu atau beberapa iterasi `Map` status gagal dijalankan atau dihentikan dalam status dibatalkan, Anda dapat melihat detail tentang iterasi yang gagal. Untuk melihat detail ini, pilih nomor iterasi yang terpengaruh di bawah Gagal atau Dibatalkan dalam daftar tarik-turun.

Tampilan tabel - Beralih di antara iterasi **`Map`** status yang berbeda

Jika status Peta Anda memiliki lima iterasi dan Anda ingin melihat detail eksekusi untuk iterasi nomor tiga dan empat, lakukan hal berikut:

1. Pilih `Map` status yang ingin Anda lihat data iterasi yang berbeda.
2. Dalam tampilan tampilan pohon iterasi `Map` status, pilih baris untuk iterasi bernama #2 untuk iterasi nomor tiga. Demikian pula, pilih baris bernama #3 untuk iterasi nomor empat.

Tampilan tabel - Konfigurasikan kolom untuk ditampilkan

Pilih

.

Kemudian, di kotak dialog Preferensi, pilih kolom yang ingin Anda tampilkan di bawah Pilih kolom yang terlihat.

Secara default, mode ini menampilkan kolom Nama, Jenis, Status, Sumber Daya, dan Dimulai Setelah.

Tampilan tabel - Filter hasilnya

Batasi jumlah informasi yang ditampilkan dengan menerapkan satu atau beberapa filter berdasarkan properti, seperti Status, atau rentang tanggal dan waktu. Misalnya, untuk melihat langkah-langkah yang gagal eksekusi, terapkan filter berikut:

1. Pilih Filter menurut properti atau cari berdasarkan kata kunci, lalu pilih Status di bawah Properti.
2. Di bawah Operator, pilih Status =.
3. Pilih Status = Gagal.
4. (Opsional) Pilih Hapus filter untuk menghapus filter yang diterapkan.

Tampilan acara - Filter hasil

Batasi jumlah informasi yang ditampilkan dengan menerapkan satu atau beberapa filter berdasarkan properti, seperti Jenis, atau rentang tanggal dan waktu. Misalnya, untuk melihat langkah-langkah `Task` status yang gagal dieksekusi, terapkan filter berikut:

1. Pilih Filter menurut properti atau cari berdasarkan kata kunci, lalu pilih Ketik di bawah Properti.
2. Di bawah Operator, pilih Type =.
3. Pilih Type = TaskFailed.
4. (Opsional) Pilih Hapus filter untuk menghapus filter yang diterapkan.

Tampilan acara - Periksa detail TaskFailedacara

Pilih di

sebelah ID TaskFailedacara untuk memeriksa detailnya, termasuk input, output, dan pemanggilan sumber daya yang muncul di kotak tarik-turun.

Redrivingeksekusi

Anda dapat menggunakan redrive untuk memulai ulang eksekusi Alur Kerja Standar yang tidak berhasil diselesaikan dalam 14 hari terakhir. Ini termasuk eksekusi yang gagal, dibatalkan, atau habis waktu.

Ketika Anda redrive eksekusi, itu melanjutkan eksekusi yang gagal dari langkah yang gagal dan menggunakan input yang sama. Step Functionsmempertahankan hasil dan riwayat eksekusi dari

langkah-langkah yang berhasil, dan langkah-langkah ini tidak dijalankan kembali saat Anda redrive mengeksekusi. Misalnya, katakan bahwa alur kerja Anda berisi dua status: Diteruskan status diikuti oleh Tugas status. Jika eksekusi alur kerja Anda gagal pada status Tugas, dan Anda redrive mengeksekusinya, eksekusi akan menjadwal ulang dan kemudian menjalankan kembali status Tugas.

Redriveneksekusi menggunakan definisi mesin keadaan yang sama dan eksekusi ARN yang digunakan untuk upaya eksekusi asli. Jika upaya eksekusi asli Anda dikaitkan dengan versi, alias, atau keduanya, redriven eksekusi dikaitkan dengan versi, alias, atau keduanya yang sama. Bahkan jika Anda memperbarui alias Anda untuk menunjuk ke versi yang berbeda, redriven eksekusi terus menggunakan versi yang terkait dengan upaya eksekusi asli. Karena redriven eksekusi menggunakan definisi mesin status yang sama, Anda harus memulai eksekusi baru jika Anda memperbarui definisi mesin status Anda.

Ketika Anda redrive eksekusi, batas waktu tingkat mesin negara, jika ditentukan, diatur ulang ke 0. Untuk informasi selengkapnya tentang batas waktu tingkat mesin negara bagian, lihat`TimeoutSeconds`.

Eksekusi redrives dianggap sebagai transisi negara. Untuk informasi tentang cara transisi status mempengaruhi penagihan, lihat Harga Step Functions.

Topik

- Redrivekelayakan untuk eksekusi yang gagal
- Redriveperilaku masing-masing negara
- Izin IAM untuk redrive eksekusi
- Redrivingeksekusi di konsol
- Redrivingeksekusi menggunakan API
- Memeriksa eksekusi redriven
- Coba lagi perilaku eksekusi redriven

Redrivekelayakan untuk eksekusi yang gagal

Anda dapat redrive mengeksekusi jika upaya eksekusi asli Anda memenuhi ketentuan berikut:

- Anda memulai eksekusi pada atau setelah 15 November 2023. Eksekusi yang Anda mulai sebelum tanggal ini tidak memenuhi syaratredrive.
- Status eksekusi tidak`SUCCEEDED`.

- Eksekusi alur kerja belum melebihi redrivable periode 14 hari. Redrivableperiode mengacu pada waktu di mana Anda dapat eksekusi redrive tertentu. Periode ini dimulai dari hari mesin negara menyelesaikan pelaksanaannya.
- Eksekusi alur kerja belum melebihi waktu buka maksimum satu tahun. Untuk informasi tentang kuota eksekusi mesin status, lihatKuota yang berkaitan dengan eksekusi mesin status.
- Jumlah riwayat acara eksekusi kurang dari 24.999. Redriveneksekusi menambahkan riwayat peristiwa mereka ke riwayat peristiwa yang ada. Pastikan eksekusi alur kerja Anda berisi kurang dari 24.999 peristiwa untuk mengakomodasi peristiwa `ExecutionRedriven` riwayat dan setidaknya satu peristiwa riwayat lainnya.

Redriveperilaku masing-masing negara

Bergantung pada status yang gagal dalam alur kerja Anda, redrive perilaku untuk semua status yang gagal bervariasi. Tabel berikut menjelaskan redrive perilaku untuk semua negara bagian.

Nama negara	Redriveperilaku eksekusi
Diteruskan	Jika langkah sebelumnya gagal atau mesin status habis, status Pass akan keluar dan tidak dieksekusi. redrive
Tugas	Menjadwalkan dan memulai status Tugas lagi. Ketika Anda eksekusi redrive yang menjalank an kembali status Tugas, `TimeoutSeconds` untuk status, jika ditentukan, disetel ulang ke 0. Untuk informasi selengkapnya tentang batas waktu, lihat Status tugas.
Pilihan	Mengevaluasi kembali aturan negara Pilihan.
Tunggu	Jika status menentukan `Timestamp` atau `TimestampPath` yang mengacu pada stempel waktu di masa lalu, redrive menyebabk an status Tunggu keluar dan memasuki status yang ditentukan di bidang. `Next`

Nama negara	Redriveperilaku eksekusi
Berhasil	Tidak redrive menyatakan eksekusi mesin yang memasuki status Sukses.
Gagal	Memasuki kembali status Gagal dan gagal lagi.
Paralel	Menjadwalkan ulang dan redrives hanya cabang-cabang yang gagal atau dibatalkan. Jika status gagal karena `States.DataLimitExceeded` kesalahan, status Paralel dijalankan kembali, termasuk cabang yang berhasil dalam upaya eksekusi asli.
Status Peta Sebaris	Menjadwalkan ulang dan redrives hanya iterasi yang gagal atau dibatalkan. Jika status gagal karena `States.DataLimitExceeded` kesalahan, status Peta Sebaris dijalankan kembali, termasuk iterasi yang berhasil dalam upaya eksekusi asli.
Status Peta Terdistribusi	redriveseksekusi alur kerja anak yang gagal dalam Map Run. Untuk informasi selengkap nya, lihat RedrivingPeta Berjalan. Jika status gagal karena `States.DataLimitExceeded` kesalahan, status Peta Terdistribusi dijalankan kembali. Ini termasuk alur kerja anak yang berhasil dalam upaya eksekusi asli.

Izin IAM untuk redrive eksekusi

Step Functions membutuhkan izin redrive yang sesuai untuk eksekusi. Contoh kebijakan IAM berikut memberikan hak istimewa paling sedikit yang diperlukan untuk mesin negara Anda untuk redriving

eksekusi. Ingatlah untuk mengganti teks yang *dicetak miring dengan informasi* spesifik sumber daya Anda.

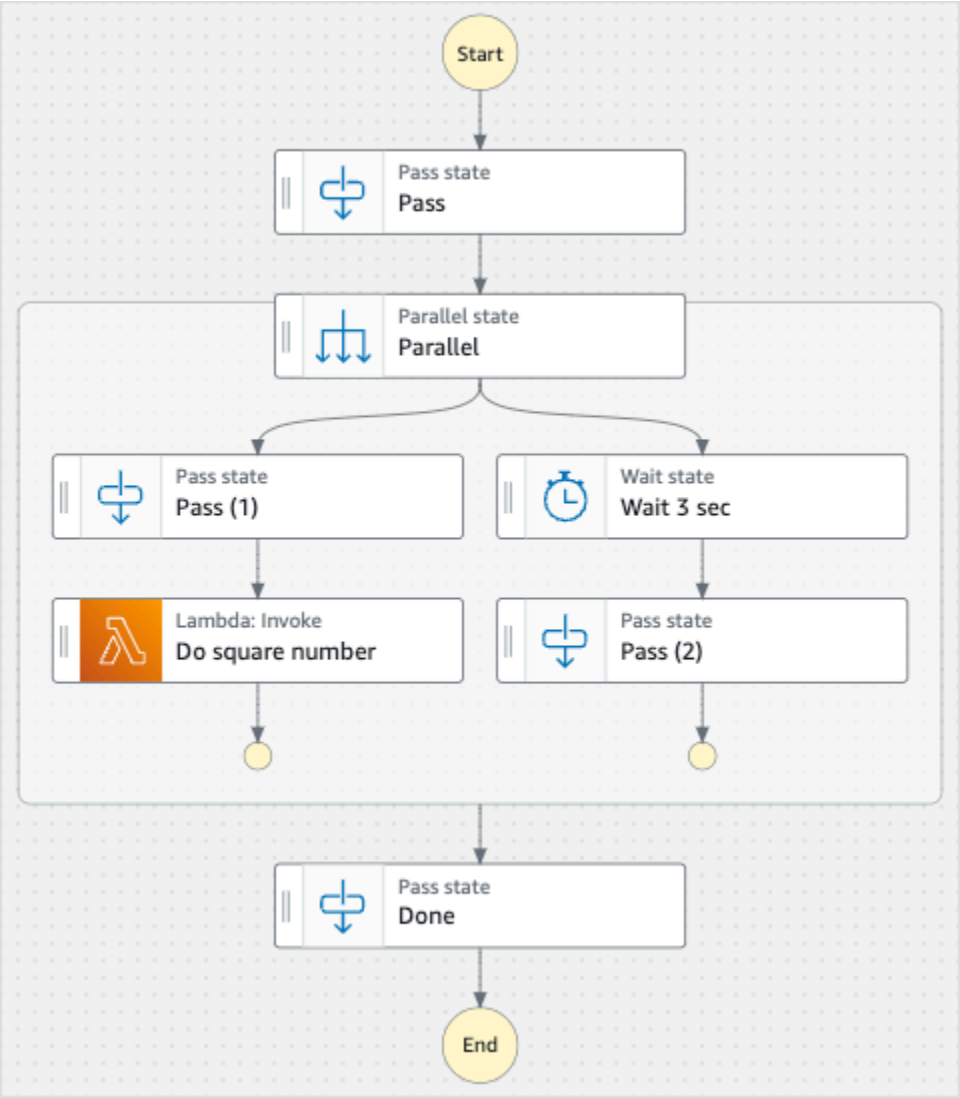
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "states:RedriveExecution"
            ],
            "Resource": "arn:aws:states:us-
east-2:123456789012:execution:myStateMachine:*"
        }
    ]
}
```

Untuk contoh izin yang Anda perlukan untuk redrive Map Run, lihatContoh kebijakan IAM untuk Peta redriving Terdistribusi.

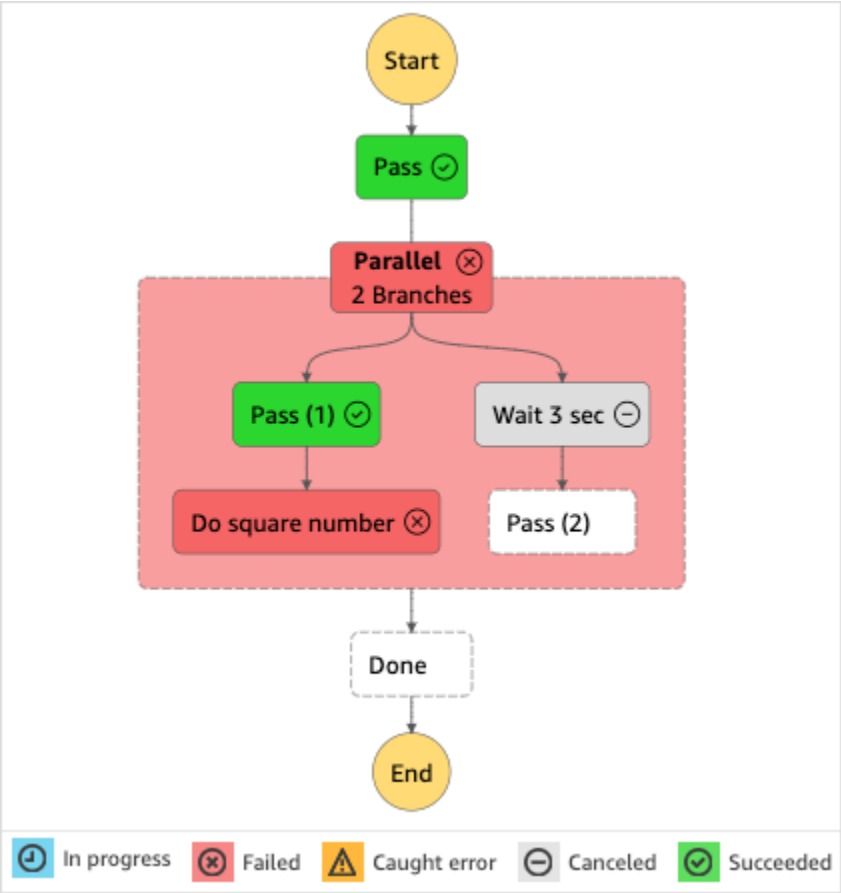
Redrivingeksekusi di konsol

Anda dapat redrive memenuhi syarat eksekusi dari Step Functions konsol.

Misalnya, katakan bahwa gambar berikut mewakili grafik alur kerja mesin status Anda.



Bayangkan Anda menjalankan mesin status ini. Gambar berikut menunjukkan grafik untuk eksekusi mesin negara.



Seperti yang ditunjukkan pada gambar ini, langkah LambdaInvoke bernama Do square number di dalam keadaan Parallel telah mengembalikan kesalahan. Hal ini menyebabkan keadaan Paralel gagal. Cabang-cabang yang eksekusinya sedang berlangsung atau tidak dimulai dihentikan dan eksekusi mesin negara gagal.

Untuk redrive eksekusi dari konsol

1. Buka konsol Step Functions, lalu pilih mesin status yang ada yang gagal eksekusi.
2. Pada halaman detail mesin status, di bawah Eksekusi, pilih instance eksekusi yang gagal.
3. Pilih Redrive.
4. Di kotak Redrivedialog, pilih Redriveeksekusi.

> **Tip**
>
> Jika Anda berada di halaman Rincian Eksekusi dari eksekusi yang gagal, lakukan salah satu hal redrive berikut untuk eksekusi:
>
> - Pilih Pulihkan, lalu pilih Redrivedari kegagalan.

- Pilih Tindakan, lalu pilih Redrive.

Perhatikan bahwa redrive menggunakan definisi mesin status yang sama dan ARN. Ini terus menjalankan eksekusi dari langkah yang gagal dalam upaya eksekusi asli. Dalam contoh ini, ini adalah langkah Do square number dan Wait 3 sec cabang di dalam keadaan Paralel. Setelah memulai ulang eksekusi langkah-langkah yang gagal ini dalam keadaan Paralel, redrive akan melanjutkan eksekusi untuk langkah Selesai.

5. Pilih eksekusi untuk membuka halaman Detail Eksekusi.

   Di halaman ini, Anda dapat melihat hasil redriven eksekusi. Misalnya, di Ringkasan eksekusi bagian ini, Anda dapat melihat Redrivehitungan, yang mewakili berapa kali eksekusi telah dilakukanredriven. Di bagian Peristiwa, Anda dapat melihat peristiwa eksekusi redrive terkait ditambahkan ke peristiwa upaya eksekusi asli. Misalnya, `ExecutionRedriven` acara.

Redrivingeksekusi menggunakan API

Anda dapat redrive memenuhi syarat eksekusi menggunakan RedriveExecutionAPI. API ini memulai ulang eksekusi Alur Kerja Standar yang tidak berhasil dari langkah yang gagal, dibatalkan, atau habis waktu.

Dalam AWS Command Line Interface (AWS CLI), jalankan perintah berikut ke eksekusi mesin status redrive yang tidak berhasil. Ingatlah untuk mengganti teks yang *`dicetak miring dengan informasi`* spesifik sumber daya Anda.

```
aws stepfunctions redrive-execution --execution-arn arn:aws:states:us-
east-2:123456789012:execution:myStateMachine:foo
```

Memeriksa eksekusi redriven

Anda dapat memeriksa redriven eksekusi di konsol atau menggunakan API: GetExecutionHistorydan DescribeExecution.

Periksa redriven eksekusi di konsol

1. Buka konsol Step Functions, lalu pilih mesin status yang sudah ada redriven yang telah Anda eksekusi.
2. Buka halaman Detail Eksekusi.

Di halaman ini, Anda dapat melihat hasil redriven eksekusi. Misalnya, di Ringkasan eksekusi bagian ini, Anda dapat melihat Redrivehitungan, yang mewakili berapa kali eksekusi telah dilakukanredriven. Di bagian Peristiwa, Anda dapat melihat peristiwa eksekusi redrive terkait ditambahkan ke peristiwa upaya eksekusi asli. Misalnya, `ExecutionRedriven` acara.

Periksa redriven eksekusi menggunakan API

Jika Anda redriven memiliki eksekusi mesin status, Anda dapat menggunakan salah satu API berikut untuk melihat detail tentang redriven eksekusi. Ingatlah untuk mengganti teks yang *`dicetak miring dengan informasi`* spesifik sumber daya Anda.

- GetExecutionHistory — Mengembalikan riwayat eksekusi yang ditentukan sebagai daftar acara. API ini juga mengembalikan detail tentang redrive upaya eksekusi, jika tersedia.

  Dalam AWS CLI, jalankan perintah berikut.

  ```
  aws stepfunctions get-execution-history --execution-arn arn:aws:states:us-
  east-2:123456789012:execution:myStateMachine:foo
  ```

- DescribeExecution — Memberikan informasi tentang eksekusi mesin negara. Ini bisa berupa mesin status yang terkait dengan eksekusi, input dan output eksekusi, redrive detail eksekusi, jika tersedia, dan metadata eksekusi yang relevan.

  Dalam AWS CLI, jalankan perintah berikut.

  ```
  aws stepfunctions describe-execution --execution-arn arn:aws:states:us-
  east-2:123456789012:execution:myStateMachine:foo
  ```

Coba lagi perilaku eksekusi redriven

Jika redriven eksekusi Anda menjalankan ulang statusTugas,Paralel, atau Peta Sebaris, yang telah Anda tetapkan percobaan ulang, jumlah upaya coba lagi untuk status ini disetel ulang ke 0. Hal ini memungkinkan untuk jumlah maksimum upaya padaredrive. Untuk redriven eksekusi, Anda dapat melacak upaya percobaan ulang individual dari status ini menggunakan konsol.

Untuk memeriksa upaya coba lagi individu di konsol

1. Pada halaman Execution Details pada konsol Step Functions, pilih status yang dicoba ulang. redrive
2. Pilih redrives tab Retries &.
3. Pilih di

   ▶

   sebelah setiap upaya coba lagi untuk melihat detailnya. Jika upaya coba lagi berhasil, Anda dapat melihat hasilnya di Output yang muncul di kotak tarik-turun.

Gambar berikut menunjukkan contoh percobaan ulang yang dilakukan untuk status dalam upaya eksekusi asli dan eksekusi itu. redrives Dalam gambar ini, tiga percobaan ulang dilakukan dalam upaya asli dan redrive eksekusi. Eksekusi berhasil dalam redrive upaya keempat dan mengembalikan output 16.

Input | Output | Details | Definition | **Retries & redrives** | Events

Type	Status	Resource	Duration	Time
▶ Original execution	Failed	Logs | Lambda | Log group	00:00:00.151	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.139	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.164	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.149	
▶ Redrive #1	Failed	Logs | Lambda | Log group	00:00:00.187	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.147	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.154	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.170	
▶ Redrive #2	Failed	Logs | Lambda | Log group	00:00:00.206	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.184	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.188	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.219	
▶ Redrive #3	Failed	Logs | Lambda | Log group	00:00:00.198	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.142	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.174	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.208	
▼ Redrive #4	Succeeded	Logs | Lambda | Log group	00:00:00.195	

Output Learn more

```
1 ▾ {
2     "Squared": 16
3   }
```

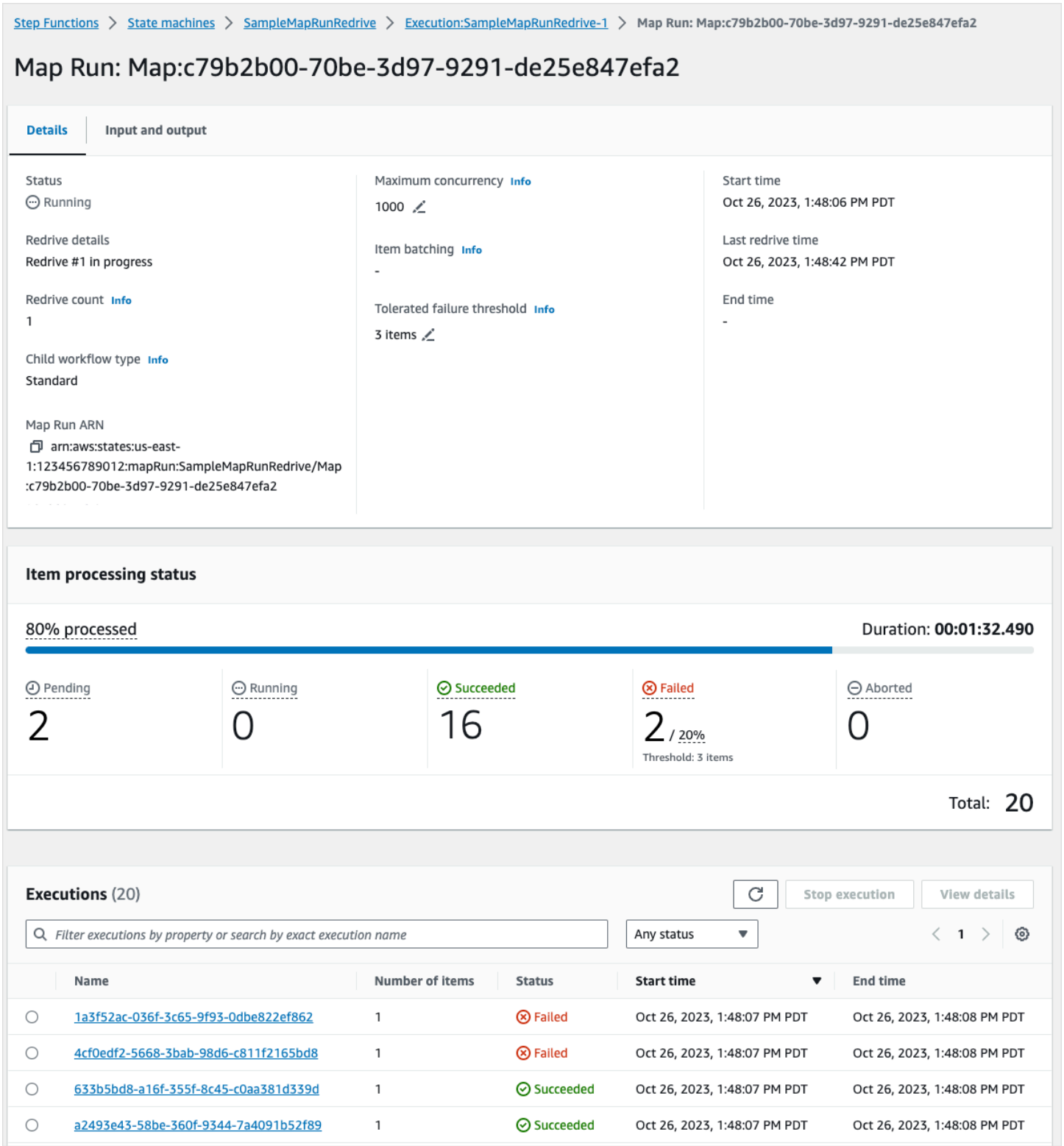
Formatted

Memeriksa Peta Jalankan eksekusi status Peta Terdistribusi

Saat Anda menjalankan `Map` status dalam mode Distributed, Step Functions akan membuat resource Map Run. Map Run mengacu pada sekumpulan eksekusi alur kerja anak yang memulai status Peta Terdistribusi, dan pengaturan runtime yang mengontrol eksekusi ini. Step Functions menetapkan Amazon Resource Name (ARN) ke Map Run Anda. Anda dapat memeriksa Map Run di konsol Step Functions. Anda juga dapat menjalankan tindakan `DescribeMapRun` API. Map Run juga memancarkan metrik ke. CloudWatch

Konsol Step Functions menyediakan halaman Map Run Details yang menampilkan semua informasi yang terkait dengan eksekusi status Peta Terdistribusi. Misalnya, Anda dapat melihat status eksekusi status Peta Terdistribusi, ARN Map Run, dan status item yang diproses dalam eksekusi alur kerja turunan yang dimulai oleh status Peta Terdistribusi. Anda juga dapat melihat daftar semua eksekusi alur kerja anak dan mengakses detailnya. Selain itu, jika Map Run Anda redriven, Anda dapat melihat redrive detail Map Run di Ringkasan eksekusi Map Run bagian tersebut. Misalnya, Terakhir redrive kali. Konsol menampilkan informasi ini dalam format dasbor.

Halaman Map Run Details berisi bagian-bagian berikut:



Daftar Isi

- Ringkasan eksekusi Map Run
- Pesan kesalahan

- Status pemrosesan item
- Daftar eksekusi
- RedrivingPeta Berjalan
  - Redrivekelayakan untuk alur kerja anak dalam Map Run
  - Perilaku eksekusi redrive alur kerja anak
  - Skenario input yang digunakan pada Map Run redrive
  - Izin IAM untuk redrive Map Run
  - RedrivingPeta Jalankan di konsol
  - RedrivingPeta Jalankan menggunakan API

Ringkasan eksekusi Map Run

Bagian ringkasan Map Run Execution muncul di bagian atas halaman Map Run Details. Bagian ini memberi Anda gambaran umum tentang detail eksekusi status Peta Terdistribusi. Informasi ini dibagi antara tab berikut:

Detail

Menampilkan informasi, seperti status eksekusi status Peta Terdistribusi, ARN Jalankan Peta, dan jenis eksekusi alur kerja anak yang dimulai oleh status Peta Terdistribusi. Anda dapat melihat konfigurasi tambahan, seperti ambang kegagalan yang ditoleransi untuk Map Run dan konkurensi maksimum yang ditentukan untuk eksekusi alur kerja anak. Anda juga dapat mengedit konfigurasi ini.

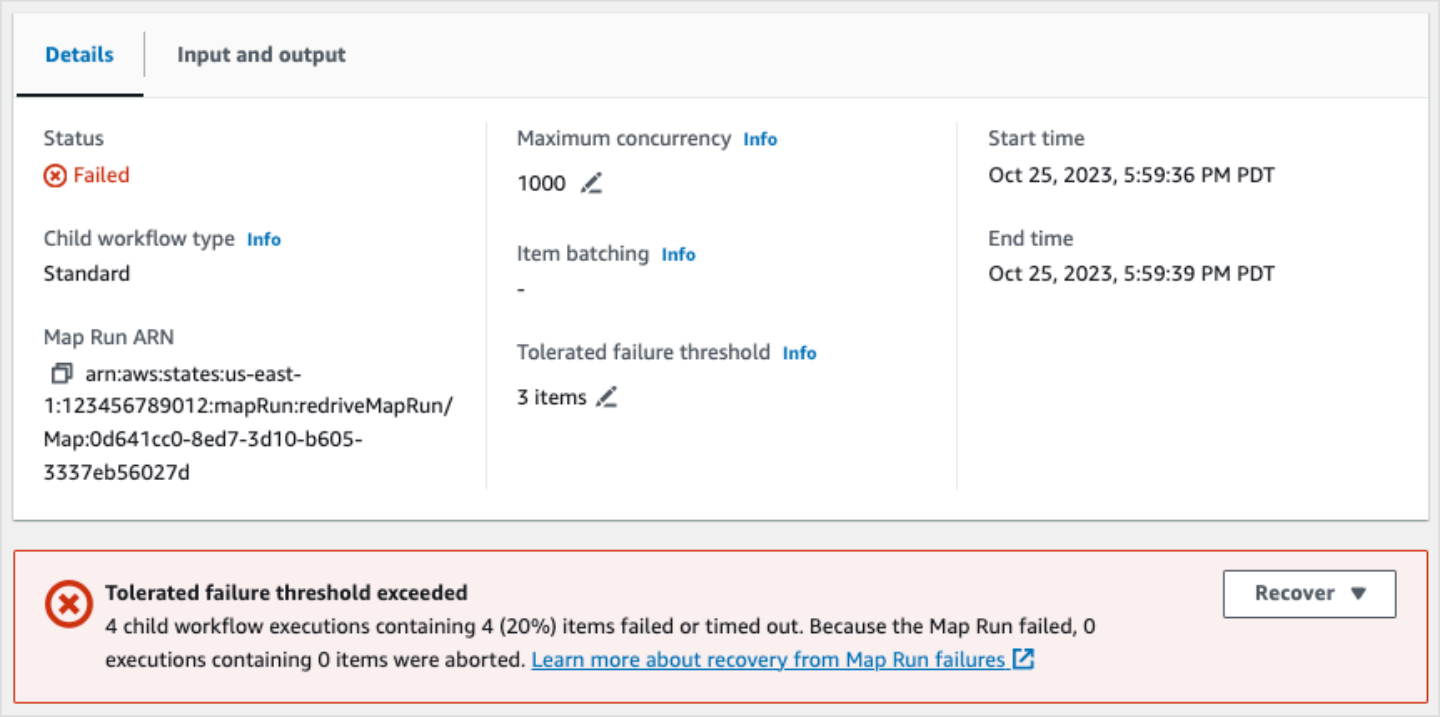
Input dan output

Menunjukkan input yang diterima oleh status Peta Terdistribusi dan output yang sesuai yang dihasilkannya. Misalnya, Anda dapat melihat kumpulan data input dan lokasinya, dan filter input diterapkan ke item data individual dalam kumpulan data tersebut. Jika Anda mengekspor output eksekusi status Peta Terdistribusi, tab ini menunjukkan jalur ke bucket Amazon S3 yang berisi hasil eksekusi. Jika tidak, ini mengarahkan Anda ke halaman Rincian Eksekusi alur kerja induk untuk melihat output eksekusi.

Pesan kesalahan

Jika Map Run gagal, halaman Map Run Details menampilkan pesan galat dengan alasan kegagalan.

Dari tombol tarik-turun Pulihkan pada pesan kesalahan ini, Anda dapat mengeksekusi alur kerja anak yang gagal dimulai oleh Map Run ini atau memulai eksekusi baru alur kerja induk. redrive Untuk informasi selengkapnya, lihat RedrivingPeta Berjalan.



Status pemrosesan item

Bagian Status pemrosesan item menampilkan status item yang diproses dalam Map Run. Misalnya, Pending menunjukkan bahwa eksekusi alur kerja anak belum mulai memproses item tersebut.

Status item tergantung pada status eksekusi alur kerja anak yang memproses item. Jika eksekusi alur kerja anak gagal, habis waktu, atau jika pengguna membatalkan eksekusi, Step Functions tidak menerima informasi apa pun tentang hasil pemrosesan item di dalam eksekusi alur kerja anak tersebut. Semua item yang diproses oleh eksekusi tersebut berbagi status eksekusi alur kerja anak.

Misalnya, Anda ingin memproses 100 item dalam dua eksekusi alur kerja anak, di mana setiap eksekusi memproses batch 50 item. Jika salah satu eksekusi gagal dan yang lainnya berhasil, Anda akan memiliki 50 item yang berhasil dan 50 gagal.

Tabel berikut menjelaskan jenis status pemrosesan yang tersedia untuk semua item:

Status	Deskripsi
Tertunda	Menunjukkan item yang eksekusi alur kerja anak belum mulai diproses. Jika Map Run berhenti, gagal, atau pengguna membatalkan eksekusi sebelum pemrosesan item dimulai, item tetap dalam status Tertunda. Misalnya, jika Map Run gagal dengan 10 item yang tertunda untuk diproses, 10 item ini tetap dalam status Tertunda.
Berlari	Menunjukkan item yang sedang diproses oleh eksekusi alur kerja anak.
Berhasil	Menunjukkan bahwa eksekusi alur kerja anak berhasil memproses item. Eksekusi alur kerja anak yang berhasil tidak dapat memiliki item yang gagal. Jika satu item dalam kumpulan data gagal selama eksekusi, seluruh eksekusi alur kerja anak gagal.
Failed	Menunjukkan bahwa eksekusi alur kerja anak gagal memproses item, atau waktu eksekusi habis. Jika salah satu item yang diproses oleh eksekusi alur kerja anak gagal, seluruh eksekusi alur kerja anak gagal. Misalnya, pertimbangkan eksekusi alur kerja anak yang memproses 1000 item. Jika salah satu item dalam kumpulan data tersebut gagal selama eksekusi, maka Step Functions menganggap seluruh eksekusi alur kerja anak gagal. Saat Anda redriveMap Run, jumlah item dengan status ini diatur ulang ke 0.

Status	Deskripsi
Dibatalkan	Menunjukkan bahwa eksekusi alur kerja anak mulai memproses item, tetapi pengguna membatalkan eksekusi, atau Step Functions menghentikan eksekusi karena Map Run gagal. Misalnya, pertimbangkan eksekusi alur kerja anak Running yang memproses 50 item. Jika Map Run berhenti karena kegagalan atau karena pengguna membatalkan eksekusi, eksekusi alur kerja anak dan status semua 50 item berubah menjadi Dibatalkan. Jika Anda menggunakan eksekusi alur kerja anak dari tipe Express, Anda tidak dapat menghentikan eksekusi. Saat Anda redriveMap Run yang memulai eksekusi alur kerja anak dari tipe Express, jumlah item dengan status ini disetel ulang ke 0. Ini karena alur kerja anak Express dimulai ulang menggunakan tindakan StartExecutionAPI alih-alih menjadi. redriven

Daftar eksekusi

Bagian Eksekusi mencantumkan semua eksekusi alur kerja anak untuk Map Run tertentu. Gunakan kolom Cari berdasarkan nama eksekusi yang tepat untuk mencari eksekusi alur kerja anak tertentu. Anda juga dapat menggunakan menu tarik-turun status apa pun untuk memfilter riwayat eksekusi alur kerja anak berdasarkan statusnya. Untuk melihat detail tentang eksekusi tertentu, pilih eksekusi alur kerja anak dari daftar dan pilih tombol Lihat detail untuk membuka halaman Rincian eksekusi.

> **Important**
>
> Kebijakan retensi untuk eksekusi alur kerja anak adalah 90 hari. Eksekusi alur kerja anak selesai yang lebih tua dari periode retensi ini tidak ditampilkan di tabel Eksekusi. Hal ini berlaku bahkan jika status Peta Terdistribusi atau alur kerja induk terus berjalan lebih lama

dari periode retensi. Anda dapat melihat detail eksekusi, termasuk hasil, dari eksekusi alur kerja turunan ini jika Anda mengekspor output status Peta Terdistribusi ke bucket Amazon S3 menggunakan. `ResultWriter`

> **Tip**
>
> Pilih tombol refresh
>
>
>
> untuk melihat daftar terbaru dari semua eksekusi alur kerja anak.

RedrivingPeta Berjalan

Anda dapat memulai ulang eksekusi alur kerja anak yang gagal di Map Run oleh alur kerja induk redrivingAnda. Alur kerja redriven induk redrives semua status gagal, termasuk Peta Terdistribusi. Alur kerja induk akan mengembalikan status yang tidak berhasil jika tidak ada `<stateType>Exited` peristiwa yang terkait dengan peristiwa untuk status saat alur kerja induk menyelesaikan eksekusinya. `<stateType>Entered` Misalnya, jika riwayat peristiwa tidak berisi peristiwa untuk suatu `MapStateExited` `MapStateEntered` peristiwa, Anda dapat redrive menjalankan alur kerja induk ke redrive semua eksekusi alur kerja anak yang gagal di Map Run.

Map Run tidak dimulai atau gagal dalam upaya eksekusi asli ketika mesin status tidak memiliki izin yang diperlukan untuk mengaksesItemReader,ResultWriter, atau keduanya. Jika Map Run tidak dimulai dalam upaya eksekusi asli alur kerja induk, alur kerja induk redriving akan memulai Map Run untuk pertama kalinya. Untuk mengembalikan ini, tambahkan izin yang diperlukan ke peran mesin status Anda, lalu redrive alur kerja induk. Jika Anda redrive alur kerja induk tanpa menambahkan izin yang diperlukan, ia mencoba untuk memulai menjalankan Map Run baru yang akan gagal lagi. Untuk informasi tentang izin yang mungkin Anda perlukan, lihatKebijakan IAM untuk menggunakan status Peta Terdistribusi.

Topik

- Redrivekelayakan untuk alur kerja anak dalam Map Run
- Perilaku eksekusi redrive alur kerja anak
- Skenario input yang digunakan pada Map Run redrive
- Izin IAM untuk redrive Map Run

- RedrivingPeta Jalankan di konsol
- RedrivingPeta Jalankan menggunakan API

Redrivekelayakan untuk alur kerja anak dalam Map Run

Anda dapat redrive mengeksekusi alur kerja anak yang gagal di Map Run jika kondisi berikut terpenuhi:

- Anda memulai eksekusi alur kerja induk pada atau setelah 15 November 2023. Eksekusi yang Anda mulai sebelum tanggal ini tidak memenuhi syaratredrive.
- Anda belum melampaui batas keras 1000 redrives dari Map Run yang diberikan. Jika Anda telah melampaui batas ini, Anda akan menerima `States.Runtime` kesalahan.
- Alur kerja induk adalahredrivable. Jika alur kerja induk tidakredrivable, Anda tidak redrive dapat mengeksekusi alur kerja anak di Map Run. Untuk informasi selengkapnya tentang redrive kelayakan alur kerja, lihat. Redrivekelayakan untuk eksekusi yang gagal
- Eksekusi alur kerja anak dari tipe Standard di Map Run Anda belum melebihi batas riwayat peristiwa eksekusi 25.000. Eksekusi alur kerja anak yang telah melampaui batas riwayat peristiwa dihitung terhadap ambang kegagalan yang ditoleransi dan dianggap gagal. Untuk informasi lebih lanjut tentang redrive kelayakan eksekusi, lihat. Redrivekelayakan untuk eksekusi yang gagal

Map Run baru dimulai dan Map Run yang ada tidak redriven dalam kasus berikut meskipun Map Run gagal dalam upaya eksekusi asli:

- Map Run gagal karena `States.DataLimitExceeded` kesalahan.
- Map Run gagal karena kesalahan interpolasi data JSON,. `States.Runtime` Misalnya, Anda memilih node JSON yang tidak ada di. OutputPath

Map Run dapat terus berjalan bahkan setelah alur kerja induk berhenti atau habis waktu. Dalam skenario ini, redrive tidak terjadi segera:

- Map Run mungkin masih membatalkan eksekusi alur kerja anak yang sedang berlangsung dari tipe Standar, atau menunggu eksekusi alur kerja anak tipe Express untuk menyelesaikan eksekusinya.
- Map Run mungkin masih menulis hasil keResultWriter, jika Anda mengonfigurasinya untuk mengekspor hasil.

Dalam kasus ini, Map Run yang sedang berjalan menyelesaikan operasinya sebelum mencoba. redrive

Perilaku eksekusi redrive alur kerja anak

Eksekusi alur kerja redriven anak di Map Run menunjukkan perilaku seperti yang dijelaskan dalam tabel berikut.

Alur kerja anak ekspres	Alur kerja anak standar
Semua eksekusi alur kerja anak yang gagal atau habis waktu dalam upaya eksekusi asli dimulai menggunakan tindakan API. StartExecution Status pertama ItemProcessor dijalankan terlebih dahulu.	Semua eksekusi alur kerja anak yang gagal, habis waktu, atau dibatalkan dalam upaya eksekusi asli redriven menggunakan tindakan API. RedriveExecution Alur kerja anak ini redriven berasal dari keadaan terakhir ItemProcessor yang mengakibatkan eksekusi mereka gagal.
Eksekusi yang gagal selalu bisa terjadi. redriven Ini karena eksekusi alur kerja anak Express selalu dimulai sebagai eksekusi baru menggunakan tindakan StartExecution API.	Eksekusi alur kerja anak standar yang tidak berhasil tidak selalu dapat dilakukan. redriven Jika eksekusi tidakredrivable, itu tidak akan dicoba lagi. Kesalahan atau output terakhir dari eksekusi bersifat permanen. Ini dimungkinkan ketika eksekusi melebihi 25.000 peristiwa sejarah, atau redrivable periode 14 hari telah kedaluwarsa. Eksekusi alur kerja anak standar mungkin tidak terjadi redrivable jika eksekusi alur kerja induk telah ditutup dalam 14 hari, tetapi eksekusi alur kerja anak ditutup lebih awal dari 14 hari.
Eksekusi alur kerja anak ekspres menggunakan ARN eksekusi yang sama dengan upaya eksekusi asli, tetapi Anda tidak dapat mengidentifikasi individu mereka dengan jelas. redrives	Eksekusi alur kerja anak standar menggunakan ARN eksekusi yang sama dengan upaya eksekusi asli. Anda dapat dengan jelas mengidentifikasi individu redrives di konsol dan menggunakan API, seperti GetExecutionHistory dan DescribeExecution. Untuk informasi

Alur kerja anak ekspres	Alur kerja anak standar
	selengkapnya, lihat Memeriksa eksekusi redriven.

Jika Anda redriven memiliki Map Run, dan telah mencapai batas konkurensinya, eksekusi alur kerja anak di Map Run itu bertransisi ke status tertunda. Status eksekusi Map Run juga bertransisi ke redrive status Pending. Sampai batas konkurensi yang ditentukan dapat memungkinkan lebih banyak eksekusi alur kerja anak berjalan, eksekusi tetap dalam status Pending. redrive

Misalnya, katakan bahwa batas konkurensi Peta Terdistribusi dalam alur kerja Anda adalah 3000, dan jumlah alur kerja anak yang akan dijalankan ulang adalah 6000. Hal ini menyebabkan 3000 alur kerja anak berjalan secara paralel sementara 3000 alur kerja sisanya tetap berada dalam status redrive Pending. Setelah batch pertama dari 3000 alur kerja anak menyelesaikan eksekusi mereka, 3000 alur kerja anak yang tersisa dijalankan.

Ketika Map Run telah menyelesaikan eksekusi atau dibatalkan, jumlah eksekusi alur kerja anak dalam redrive status Tertunda diatur ulang ke 0.

Skenario input yang digunakan pada Map Run redrive

Bergantung pada cara Anda memberikan masukan ke Peta Terdistribusi dalam upaya eksekusi asli, redriven Map Run akan menggunakan input seperti yang dijelaskan dalam tabel berikut.

Masukan dalam upaya eksekusi asli	Masukan yang digunakan pada Map Run redrive
Masukan diteruskan dari status sebelumnya atau input eksekusi.	redrivenMap Run menggunakan input yang sama.
Input diteruskan menggunakan ItemReader dan Map Run tidak memulai eksekusi alur kerja anak karena salah satu kondisi berikut benar: • Map Run gagal dengan `States.ItemReaderFailed` kesalahan. • Map Run gagal dengan `States.ResultWriterFailed` kesalahan.	redrivenMap Run menggunakan input di bucket Amazon S3.

Masukan dalam upaya eksekusi asli	Masukan yang digunakan pada Map Run redrive
• Eksekusi alur kerja induk telah habis waktu atau dibatalkan sebelum Map Run dimulai.	
Masukan diteruskan menggunakan ItemReader. Map Run gagal setelah memulai atau mencoba memulai eksekusi alur kerja anak.	redrivenMap Run menggunakan input yang sama yang disediakan dalam upaya eksekusi asli.

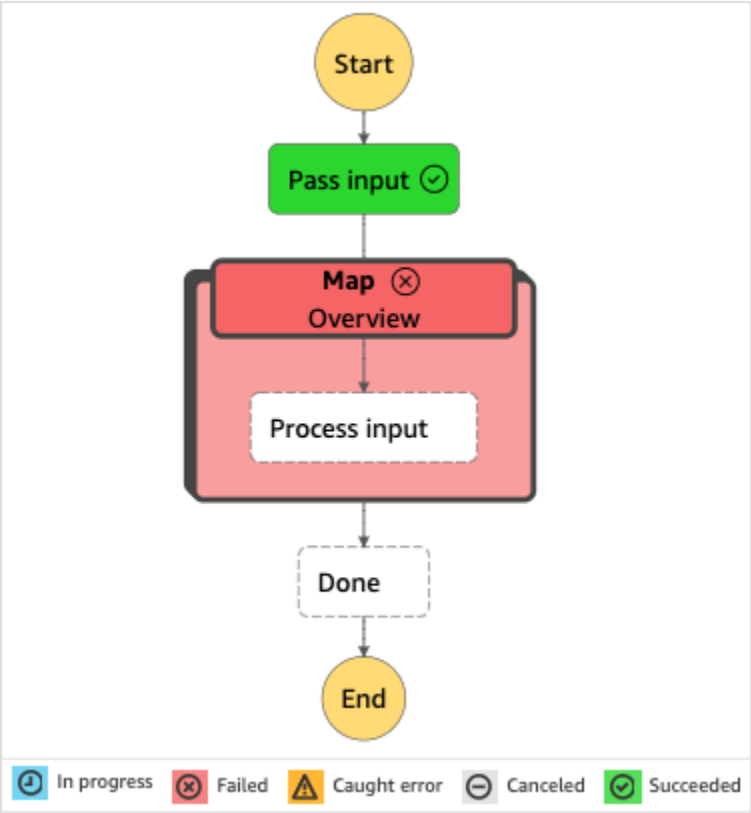
Izin IAM untuk redrive Map Run

Step Functions membutuhkan izin redrive yang sesuai untuk Map Run. Contoh kebijakan IAM berikut memberikan hak istimewa paling sedikit yang diperlukan untuk mesin status Anda untuk Map Run. redriving Ingatlah untuk mengganti teks yang *dicetak miring dengan informasi* spesifik sumber daya Anda.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:RedriveExecution"
      ],
      "Resource": "arn:aws:states:us-
east-2:123456789012:execution:myStateMachine/myMapRunLabel:*"
    }
  ]
}
```

RedrivingPeta Jalankan di konsol

Gambar berikut menunjukkan grafik eksekusi mesin negara yang berisi Peta Terdistribusi. Eksekusi ini gagal karena Map Run gagal. Untuk redrive Map Run, Anda harus redrive alur kerja induk.



Ke redrive Map Run dari konsol

1. Buka konsol Step Functions, lalu pilih mesin status yang ada yang berisi Peta Terdistribusi yang gagal dieksekusi.
2. Pada halaman detail mesin status, di bawah Eksekusi, pilih instance eksekusi gagal dari mesin status ini.
3. Pilih Redrive.
4. Di kotak Redrivedialog, pilih Redriveeksekusi.

> **Tip**
>
> Anda juga dapat menjalankan redrive peta dari halaman Rincian Eksekusi atau Rincian Jalankan Peta.
> Jika Anda berada di halaman Detail Eksekusi, lakukan salah satu hal redrive berikut untuk eksekusi:
>
> - Pilih Pulihkan, lalu pilih Redrivedari kegagalan.
> - Pilih Tindakan, lalu pilih Redrive.

Jika Anda berada di halaman Map Run Details, pilih Recover, lalu pilih Redrivedari kegagalan.

Perhatikan bahwa redrive menggunakan definisi mesin status yang sama dan ARN. Ini terus menjalankan eksekusi dari langkah yang gagal dalam upaya eksekusi asli. Dalam contoh ini, ini adalah langkah Peta Terdistribusi bernama Map dan langkah input Proses di dalamnya. Setelah memulai ulang eksekusi alur kerja anak yang gagal dari Map Run, redrive akan melanjutkan eksekusi untuk langkah Selesai.

5. Dari halaman Detail Eksekusi, pilih Map Run untuk melihat detail redriven Map Run.

   Di halaman ini, Anda dapat melihat hasil redriven eksekusi. Misalnya, di Ringkasan eksekusi Map Run bagian ini, Anda dapat melihat Redrivehitungan, yang mewakili berapa kali Map Run telahredriven. Di bagian Peristiwa, Anda dapat melihat peristiwa eksekusi redrive terkait ditambahkan ke peristiwa upaya eksekusi asli. Misalnya, `MapRunRedriven` acara.

Setelah Map Run, Anda dapat memeriksa redrive detailnya di konsol atau menggunakan tindakan DescribeExecutionAPI GetExecutionHistorydan. redriven Untuk informasi selengkapnya tentang memeriksa redriven eksekusi, lihatMemeriksa eksekusi redriven.

RedrivingPeta Jalankan menggunakan API

Anda dapat redrive menjalankan Map Run yang memenuhi syarat menggunakan RedriveExecutionAPI pada alur kerja induk. API ini memulai ulang eksekusi alur kerja anak yang gagal di Map Run.

Dalam AWS Command Line Interface (AWS CLI), jalankan perintah berikut ke eksekusi mesin status redrive yang tidak berhasil. Ingatlah untuk mengganti teks yang *dicetak miring dengan informasi* spesifik sumber daya Anda.

```
aws stepfunctions redrive-execution --execution-arn arn:aws:states:us-
east-2:123456789012:execution:myStateMachine:foo
```

Setelah Map Run, Anda dapat memeriksa redrive detailnya di konsol atau menggunakan tindakan DescribeMapRunAPI. redriven Untuk memeriksa redrive detail eksekusi alur kerja Standar di Map Run, Anda dapat menggunakan tindakan GetExecutionHistoryatau DescribeExecutionAPI. Untuk

informasi selengkapnya tentang memeriksa redriven eksekusi, lihatthe section called “Memeriksa eksekusi redriven”.

Anda dapat memeriksa redrive detail eksekusi alur kerja Express di Map Run di konsol Step Functions jika Anda mengaktifkan logging pada alur kerja induk. Untuk informasi selengkapnya, lihat Logging menggunakanCloudWatchLog.

Penanganan kesalahan dalam Step Functions

Semua status, kecuali `Pass` dan `Wait` status, dapat mengalami kesalahan runtime. Kesalahan dapat terjadi karena berbagai alasan, seperti contoh berikut:

- Masalah ketentuan mesin status (misalnya, tidak ada aturan yang cocok dalam status `Choice`)
- Kegagalan tugas (misalnya, pengecualian dalam suatu AWS Lambda fungsi)
- Masalah transien (misalnya, peristiwa partisi jaringan)

Secara default, ketika keadaan melaporkan kesalahan, AWS Step Functions menyebabkan eksekusi gagal sepenuhnya.

> **Tip**
>
> Untuk menerapkan contoh alur kerja yang menyertakan penanganan kesalahan ke AndaAkun AWS, lihat Modul 8 - Penanganan Kesalahan Lokakarya. AWS Step Functions

Topik

- Nama kesalahan
- Mencoba kembali setelah kesalahan
- Status fallback
- Nyatakan contoh mesin menggunakan Coba Ulang dan menggunakan Catch

Nama kesalahan

Step Functions mengidentifikasi kesalahan dalam Bahasa Status Amazon menggunakan string peka huruf besar-kecil, yang dikenal sebagai nama kesalahan. Bahasa Status Amazon menentukan satu set string built-in yang menamai kesalahan terkenal, semua dimulai dengan prefiks `States.`.

`States.ALL`

Sebuah wildcard yang cocok dengan nama kesalahan yang dikenal.

> **Note**
>
> Jenis kesalahan ini tidak dapat menangkap jenis kesalahan `States.DataLimitExceeded` terminal dan jenis kesalahan runtime. Untuk informasi selengkapnya tentang jenis kesalahan ini, lihat `States.DataLimitExceeded`dan `States.Runtime`.

`States.DataLimitExceeded`

Step Functions melaporkan `States.DataLimitExceeded` pengecualian dalam kondisi berikut:

- Ketika output konektor lebih besar dari kuota ukuran muatan.
- Ketika output status lebih besar dari kuota ukuran muatan.
- Ketika, setelah pengolahan `Parameters`, input status lebih besar dari kuota ukuran muatan.

Untuk informasi lebih lanjut tentang kuota, lihat Kuota.

> **Note**
>
> Ini adalah kesalahan terminal yang tidak dapat ditangkap oleh jenis `States.ALL` kesalahan.

`States.ExceedToleratedFailureThreshold`

`Map`Status gagal karena jumlah item yang gagal melebihi ambang batas yang ditentukan dalam definisi mesin status. Untuk informasi selengkapnya, lihat Ambang kegagalan yang ditoleransi untuk status Peta Terdistribusi.

`States.HeartbeatTimeout`

Sebuah `Task` negara gagal mengirim detak jantung untuk jangka waktu yang lebih lama dari `HeartbeatSeconds` nilainya.

> **Note**
>
> Kesalahan ini hanya tersedia di dalam `Catch` dan `Retry` bidang.

`States.IntrinsicFailure`

Upaya untuk menjalankan fungsi intrinsik dalam template payload gagal.

`States.ItemReaderFailed`

`Map`Status gagal karena tidak dapat membaca dari sumber item yang ditentukan di `ItemReader` bidang. Untuk informasi selengkapnya, lihat `ItemReader`.

`States.NoChoiceMatched`

`Choice`Status gagal mencocokkan input dengan kondisi yang ditentukan dalam Aturan Pilihan dan transisi Default tidak ditentukan.

`States.ParameterPathFailure`

Upaya untuk mengganti bidang, dalam `Parameters` bidang status, yang namanya berakhir `.$` dengan menggunakan jalur gagal.

`States.Permissions`

`Task`Status gagal karena memiliki hak istimewa yang tidak memadai untuk menjalankan kode yang ditentukan.

`States.ResultPathMatchFailure`

Step Functions gagal menerapkan `ResultPath` bidang status ke input status yang diterima.

`States.ResultWriterFailed`

`Map`Status gagal karena tidak dapat menulis hasil ke tujuan yang ditentukan di `ResultWriter` bidang. Untuk informasi selengkapnya, lihat `ResultWriter`.

`States.Runtime`

Eksekusi gagal karena beberapa pengecualian yang tidak dapat diproses. Sering kali ini disebabkan oleh kesalahan saat waktu aktif, seperti mencoba untuk menerapkan `InputPath` atau `OutputPath` pada muatan JSON null. `States.Runtime`Kesalahan tidak dapat diambil

kembali, dan akan selalu menyebabkan eksekusi gagal. Sebuah retry atau catch on `States.ALL` won't catch `States.Runtime` error.

`States.TaskFailed`

Status `Task` gagal selama eksekusi. Ketika digunakan dalam coba lagi atau menangkap, `States.TaskFailed` bertindak sebagai wildcard yang cocok dengan nama kesalahan yang diketahui kecuali untuk `States.Timeout`.

`States.Timeout`

Status `Task` berjalan lebih lama daripada nilai `TimeoutSeconds`, atau gagal untuk mengirim detak jantung untuk jangka waktu lebih lama dari nilai `HeartbeatSeconds`.

Selain itu, jika mesin status berjalan lebih lama dari `TimeoutSeconds` nilai yang ditentukan, eksekusi gagal dengan `States.Timeout` kesalahan.

Status dapat melaporkan kesalahan dengan nama lain. Namun, nama kesalahan ini tidak dapat dimulai dengan `States.` awalan.

Sebagai praktik terbaik, pastikan kode produksi dapat menangani pengecualian layanan AWS Lambda (`Lambda.ServiceException` dan `Lambda.SdkClientException`). Untuk informasi selengkapnya, lihat Menangani pengecualian layanan Lambda.

> **Note**
>
> Kesalahan tidak tertangani di Lambda dilaporkan sebagai `Lambda.Unknown` dalam output kesalahan. Ini termasuk out-of-memory kesalahan dan batas waktu fungsi. Anda dapat mencocokkan di `Lambda.Unknown`, `States.ALL`, atau `States.TaskFailed` untuk menangani kesalahan ini. Ketika Lambda mencapai jumlah maksimum permintaan, kesalahannya adalah `Lambda.TooManyRequestsException`. Untuk informasi selengkapnya tentang kesalahan fungsi Lambda, lihat Penanganan kesalahan dan percobaan ulang otomatis di Panduan PengembangAWS Lambda.

Mencoba kembali setelah kesalahan

`Task`,`Parallel`, dan `Map` status dapat memiliki bidang bernama`Retry`, yang nilainya harus berupa array objek yang dikenal sebagai retrier. Pencoba ulang individu mewakili sejumlah percobaan kembali, biasanya pada interval waktu yang meningkat.

Ketika salah satu status ini melaporkan kesalahan dan ada `Retry` bidang, Step Functions memindai melalui retrier dalam urutan yang tercantum dalam array. Ketika nama kesalahan muncul di nilai `ErrorEquals` bidang retrier, mesin status melakukan upaya coba lagi seperti yang didefinisikan di `Retry` bidang.

Jika redriven eksekusi Anda menjalankan ulang statusTugas,Paralel, atau Peta Sebaris, yang telah Anda tetapkan percobaan ulang, jumlah upaya coba lagi untuk status ini diatur ulang ke 0 untuk memungkinkan jumlah upaya maksimum. redrive Untuk redriven eksekusi, Anda dapat melacak upaya percobaan ulang individual dari status ini menggunakan konsol. Untuk informasi selengkapnya, lihat Coba lagi perilaku eksekusi redriven di Redrivingeksekusi.

Retrier berisi bidang-bidang berikut:

> **Note**
>
> Percobaan ulang diperlakukan sebagai transisi status. Untuk informasi tentang cara transisi status mempengaruhi penagihan, lihat Harga Step Functions.

`ErrorEquals` (Wajib)

Sebuah array string nonkosong yang cocok dengan nama kesalahan. Ketika status melaporkan kesalahan, Step Functions memindai melalui retrier. Ketika nama kesalahan muncul dalam array ini, array menerapkan kebijakan coba lagi yang dijelaskan dalam retier ini.

`IntervalSeconds` (Opsional)

Sebuah bilangan bulat positif yang mewakili jumlah detik sebelum percobaan ulang pertama (secara `1` default). `IntervalSeconds`memiliki nilai maksimum 99999999.

`MaxAttempts` (Opsional)

Sebuah bilangan bulat positif yang mewakili jumlah maksimum percobaan coba lagi (`3` secara default). Jika kesalahan berulang kali lebih dari yang ditentukan, percobaan ulang berhenti dan penanganan kesalahan normal dilanjutkan. Nilai `0` menentukan bahwa kesalahan tidak pernah dicoba lagi. `MaxAttempts`memiliki nilai maksimum 99999999.

`BackoffRate` (Opsional)

Pengganda dimana interval coba lagi dilambangkan dengan `IntervalSeconds` peningkatan setelah setiap upaya coba lagi. Secara default, `BackoffRate` nilainya meningkat sebesar`2.0`.

Misalnya, katakanlah Anda `IntervalSeconds` adalah 3, `MaxAttempts` adalah 3, dan `BackoffRate` 2. Upaya coba lagi pertama dilakukan tiga detik setelah kesalahan terjadi. Percobaan kedua berlangsung enam detik setelah upaya coba lagi pertama. Sedangkan percobaan ulang ketiga berlangsung 12 detik setelah upaya coba lagi kedua.

`MaxDelaySeconds` (Opsional)

Bilangan bulat positif yang menetapkan nilai maksimum, dalam hitungan detik, hingga interval coba lagi dapat meningkat. Bidang ini berguna untuk digunakan dengan `BackoffRate` bidang. Nilai yang Anda tentukan di bidang ini membatasi waktu tunggu eksponensial yang dihasilkan dari pengganda tingkat backoff yang diterapkan pada setiap upaya percobaan ulang berturut-turut. Anda harus menentukan nilai yang lebih besar dari 0 dan kurang dari 31622401 untuk. `MaxDelaySeconds`

Jika Anda tidak menentukan nilai ini, Step Functions tidak membatasi waktu tunggu antara percobaan ulang.

`JitterStrategy` (Opsional)

String yang menentukan apakah akan menyertakan jitter dalam waktu tunggu antara upaya coba lagi berturut-turut. Jitter mengurangi upaya percobaan ulang simultan dengan menyebarkannya melalui interval penundaan acak. String ini menerima `FULL` atau `NONE` sebagai nilainya. Nilai bawaannya adalah `NONE`.

Misalnya, Anda telah menetapkan `MaxAttempts` sebagai 3, `IntervalSeconds` sebagai 2, dan `BackoffRate` sebagai 2. Upaya coba lagi pertama dilakukan dua detik setelah kesalahan terjadi. Percobaan kedua berlangsung empat detik setelah percobaan ulang pertama dan percobaan ulang ketiga berlangsung delapan detik setelah upaya coba lagi kedua. Jika Anda menetapkan `JitterStrategy` sebagai`FULL`, interval coba ulang pertama diacak antara 0 dan 2 detik, interval coba lagi kedua diacak antara 0 dan 4 detik, dan interval coba ulang ketiga diacak antara 0 dan 8 detik.

Coba lagi contoh bidang

Bagian ini mencakup contoh `Retry` bidang berikut.

- Retry with BackoffRate
- Retry with MaxDelaySeconds
- Retry all errors except States.Timeout
- Complex retry scenario

> **Tip**
>
> Untuk menerapkan contoh alur kerja penanganan kesalahan ke AndaAkun AWS, lihat modul Penanganan Kesalahan dari Lokakarya. AWS Step Functions

Contoh 1 — Coba lagi dengan BackoffRate

Contoh berikut ini `Retry` membuat dua percobaan lagi dengan percobaan ulang pertama terjadi setelah menunggu selama tiga detik. Berdasarkan yang `BackoffRate` Anda tentukan, Step Functions meningkatkan interval antara setiap percobaan ulang hingga jumlah maksimum percobaan ulang tercapai. Dalam contoh berikut, upaya coba lagi kedua dimulai setelah menunggu selama tiga detik setelah percobaan ulang pertama.

```
"Retry": [ {
   "ErrorEquals": [ "States.Timeout" ],
   "IntervalSeconds": 3,
   "MaxAttempts": 2,
   "BackoffRate": 1
} ]
```

Contoh 2 — Coba lagi dengan MaxDelaySeconds

Contoh berikut membuat tiga upaya coba lagi dan membatasi waktu tunggu yang dihasilkan dari `BackoffRate` 5 detik. Coba lagi pertama dilakukan setelah menunggu selama tiga detik. Upaya coba lagi kedua dan ketiga terjadi setelah menunggu selama lima detik setelah upaya coba lagi sebelumnya karena batas waktu tunggu maksimum yang ditetapkan oleh. `MaxDelaySeconds`

```
"Retry": [ {
    "ErrorEquals": [ "States.Timeout" ],
    "IntervalSeconds": 3,
    "MaxAttempts": 3,
    "BackoffRate":2,
    "MaxDelaySeconds": 5,
    "JitterStrategy": "FULL"
} ]
```

Tanpa`MaxDelaySeconds`, upaya coba lagi kedua akan berlangsung enam detik setelah percobaan ulang pertama, dan upaya coba lagi ketiga akan berlangsung 12 detik setelah percobaan ulang kedua.

Contoh 3 - Coba lagi semua kesalahan kecuali States.Timeout

Nama terpesan `States.ALL` yang muncul di bidang `ErrorEquals` adalah wildcard yang cocok dengan setiap nama kesalahan. Ini harus muncul sendirian di array `ErrorEquals` dan harus muncul dalam retrier terakhir di array `Retry`. Nama `States.TaskFailed` juga bertindak wildcard dan cocok dengan kesalahan kecuali untuk`States.Timeout`.

Contoh `Retry` bidang berikut mencoba ulang kesalahan apa pun kecuali`States.Timeout`.

```
"Retry": [ {
   "ErrorEquals": [ "States.Timeout" ],
   "MaxAttempts": 0
}, {
   "ErrorEquals": [ "States.ALL" ]
} ]
```

Contoh 4 - Skenario coba lagi yang kompleks

Parameter retrier A ini berlaku di semua kunjungan ke retrier dalam konteks eksekusi status tunggal.

Pertimbangkan status `Task` berikut.

```
"X": {
   "Type": "Task",
   "Resource": "arn:aws:states:us-east-1:123456789012:task:X",
   "Next": "Y",
   "Retry": [ {
      "ErrorEquals": [ "ErrorA", "ErrorB" ],
      "IntervalSeconds": 1,
      "BackoffRate": 2.0,
      "MaxAttempts": 2
   }, {
      "ErrorEquals": [ "ErrorC" ],
      "IntervalSeconds": 5
   } ],
   "Catch": [ {
      "ErrorEquals": [ "States.ALL" ],
      "Next": "Z"
   } ]
}
```

Tugas ini gagal empat kali berturut-turut, mengeluarkan nama kesalahan ini:`ErrorA`,, `ErrorB``ErrorC`, dan. `ErrorB` Berikut ini terjadi sebagai hasilnya:

- Dua kesalahan pertama cocok dengan retrier pertama dan menyebabkan menunggu satu dan dua detik.
- Kesalahan ketiga cocok dengan retrier kedua dan menyebabkan menunggu lima detik.
- Kesalahan keempat juga cocok dengan retrier pertama. Namun, itu sudah mencapai maksimum dua percobaan ulang (`MaxAttempts`) untuk kesalahan tertentu. Oleh karena itu, retrier itu gagal dan eksekusi mengalihkan alur kerja ke `Z` status melalui bidang. `Catch`

Status fallback

`Task`, `Map` dan `Parallel` negara masing-masing dapat memiliki bidang bernama`Catch`. Nilai bidang ini harus berupa array obyek, yang dikenal sebagai penangkap.

Penangkap berisi bidang berikut.

`ErrorEquals` (Wajib)

Sebuah array string non-kosong yang cocok dengan nama kesalahan, ditentukan persis seperti mereka dengan bidang retrier dengan nama yang sama.

`Next` (Wajib)

Sebuah string yang harus sama persis dengan salah satu nama status mesin status.

`ResultPath` (Opsional)

Jalur yang menentukan input apa yang dikirimkan oleh penangkap ke status yang ditentukan di `Next` bidang.

Ketika status melaporkan kesalahan dan tidak ada bidang `Retry`, atau jika mencoba ulang gagal untuk menyelesaikan kesalahan, Step Functions memindai penangkap dalam urutan yang tercantum dalam array. Ketika nama kesalahan muncul dalam nilai bidang `ErrorEquals` penangkap, mesin status bertransisi ke status bernama dalam bidang `Next`.

Nama terpesan `States.ALL` yang muncul di bidang `ErrorEquals` penangkap adalah wildcard yang cocok dengan nama kesalahan. Ini harus muncul sendirian di array `ErrorEquals` dan harus muncul di penangkap terakhir di array `Catch`. Nama `States.TaskFailed` juga bertindak wildcard dan cocok dengan kesalahan kecuali untuk `States.Timeout`.

Contoh berikut dari bidang transisi `Catch` ke status bernama `RecoveryState` ketika fungsi Lambda meng-output pengecualian Java yang tidak tertangani. Jika tidak, bidang bertransisi ke status `EndState`.

```
"Catch": [ {
   "ErrorEquals": [ "java.lang.Exception" ],
   "ResultPath": "$.error-info",
   "Next": "RecoveryState"
}, {
   "ErrorEquals": [ "States.ALL" ],
   "Next": "EndState"
} ]
```

> **Note**
>
> Setiap penangkap dapat menentukan beberapa kesalahan untuk menangani.

Output kesalahan

Ketika Step Functions bertransisi ke status yang ditentukan dalam nama penangkapan, objek biasanya berisi bidang `Cause`. Nilai bidang ini adalah deskripsi yang dapat dibaca manusia dari kesalahan. Objek ini dikenal sebagai output kesalahan.

Dalam contoh ini, penangkap pertama berisi bidang `ResultPath`. Ini bekerja sama dengan `ResultPath` di tingkat atas status, menghasilkan dua kemungkinan:

- Dibutuhkan hasil eksekusi negara itu dan menimpa semua, atau sebagian, masukan negara.
- Dibutuhkan hasil dan menambahkannya ke input. Dalam kasus kesalahan yang ditangani oleh penangkap, hasil eksekusi status adalah output kesalahan.

Jadi, untuk penangkap pertama dalam contoh, penangkap menambahkan output kesalahan ke input sebagai bidang bernama `error-info` jika belum ada bidang dengan nama ini di input. Kemudian, penangkap mengirimkan seluruh input ke`RecoveryState`. Untuk penangkap kedua, output kesalahan menimpa input dan penangkap hanya mengirimkan output kesalahan ke. `EndState`

> **Note**
>
> Jika Anda tidak menentukan bidang `ResultPath`, bidang tersebut default ke `$`, yang memilih dan menimpa seluruh input.

Ketika status memiliki keduanya `Retry` dan `Catch` bidang, Step Functions menggunakan retrier yang sesuai terlebih dahulu. Jika kebijakan coba lagi gagal menyelesaikan kesalahan, Step Functions akan menerapkan transisi penangkap yang cocok.

Muatan penyebab dan integrasi layanan

Sebuah penangkap mengembalikan muatan string sebagai output. Ketika bekerja dengan integrasi layanan seperti Amazon Athena atau AWS CodeBuild, Anda mungkin ingin mengonversi string `Cause` ke JSON. Contoh berikut dari `Pass` keadaan dengan fungsi intrinsik menunjukkan bagaimana untuk mengkonversi `Cause` string ke JSON.

```
"Handle escaped JSON with JSONtoString": {
  "Type": "Pass",
  "Parameters": {
    "Cause.$": "States.StringToJson($.Cause)"
  },
  "Next": "Pass State with Pass Processing"
},
```

Nyatakan contoh mesin menggunakan Coba Ulang dan menggunakan Catch

Mesin-mesin status yang ditentukan dalam contoh berikut memakai dua fungsi Lambda: salah satu yang selalu gagal dan satu yang menunggu cukup lama untuk mengizinkan batas waktu yang ditentukan dalam mesin status terjadi.

Ini adalah definisi dari fungsi Lambda Node.js yang selalu gagal, mengembalikan pesan. `error` Dalam contoh mesin status yang mengikuti, fungsi Lambda ini bernama `FailFunction`. Untuk informasi tentang membuat fungsi Lambda, lihat Langkah 1: Membuat fungsi Lambda bagian.

```
exports.handler = (event, context, callback) => {
    callback("error");
};
```

Ini adalah definisi dari fungsi Lambda Node.js yang tidur selama 10 detik. Dalam contoh mesin status yang mengikuti, fungsi Lambda ini bernama `sleep10`.

> **Note**
>
> Bila Anda membuat fungsi Lambda ini di konsol Lambda, ingat untuk mengubah nilai Waktu habis dalam bagian Pengaturan lanjutan dari 3 detik (default) hingga 11 detik.

```
exports.handler = (event, context, callback) => {
    setTimeout(function(){
    }, 11000);
};
```

Menangani kegagalan menggunakan Coba Lagi

Mesin status ini menggunakan `Retry` untuk mencoba kembali fungsi yang gagal dan output nama kesalahan `HandledError`. Ini mencoba ulang fungsi ini dua kali dengan backoff eksponensial antara percobaan ulang.

```
{
   "Comment": "A Hello World example of the Amazon States Language using an AWS Lambda
 function",
   "StartAt": "HelloWorld",
   "States": {
      "HelloWorld": {
         "Type": "Task",
         "Resource": "arn:aws:lambda:us-east-1:123456789012:function:FailFunction",
         "Retry": [ {
            "ErrorEquals": ["HandledError"],
            "IntervalSeconds": 1,
            "MaxAttempts": 2,
            "BackoffRate": 2.0
         } ],
      "End": true
      }
   }
}
```

Varian ini menggunakan kode kesalahan yang telah ditentukan `States.TaskFailed`, yang cocok dengan kesalahan yang fungsi Lambda output.

```
{
   "Comment": "A Hello World example of the Amazon States Language using an AWS Lambda
 function",
   "StartAt": "HelloWorld",
   "States": {
      "HelloWorld": {
         "Type": "Task",
         "Resource": "arn:aws:lambda:us-east-1:123456789012:function:FailFunction",
         "Retry": [ {
            "ErrorEquals": ["States.TaskFailed"],
            "IntervalSeconds": 1,
            "MaxAttempts": 2,
            "BackoffRate": 2.0
         } ],
         "End": true
      }
   }
}
```

> **Note**
>
> Sebagai praktik terbaik, tugas-tugas yang referensi fungsi Lambda harus menangani pengecualian layanan Lambda. Untuk informasi selengkapnya, lihat Menangani pengecualian layanan Lambda.

Menangani kegagalan menggunakan Catch

Contoh ini menggunakan bidang `Catch`. Ketika fungsi Lambda mengeluarkan kesalahan, ia menangkap kesalahan dan mesin status bertransisi ke status. `fallback`

```
{
   "Comment": "A Hello World example of the Amazon States Language using an AWS Lambda
 function",
   "StartAt": "HelloWorld",
   "States": {
      "HelloWorld": {
         "Type": "Task",
         "Resource": "arn:aws:lambda:us-east-1:123456789012:function:FailFunction",
         "Catch": [ {
            "ErrorEquals": ["HandledError"],
```

```
            "Next": "fallback"
         } ],
         "End": true
      },
      "fallback": {
         "Type": "Pass",
         "Result": "Hello, AWS Step Functions!",
         "End": true
      }
   }
}
```

Varian ini menggunakan kode kesalahan yang telah ditetapkan `States.TaskFailed`, yang cocok dengan kesalahan yang fungsi Lambda output.

```
{
   "Comment": "A Hello World example of the Amazon States Language using an AWS Lambda
 function",
   "StartAt": "HelloWorld",
   "States": {
      "HelloWorld": {
         "Type": "Task",
         "Resource": "arn:aws:lambda:us-east-1:123456789012:function:FailFunction",
         "Catch": [ {
            "ErrorEquals": ["States.TaskFailed"],
            "Next": "fallback"
         } ],
      "End": true
      },
      "fallback": {
         "Type": "Pass",
         "Result": "Hello, AWS Step Functions!",
         "End": true
      }
   }
}
```

Menangani batas waktu menggunakan Retry

Mesin status ini menggunakan `Retry` bidang untuk mencoba kembali `Task` status yang habis waktu, berdasarkan nilai batas waktu yang ditentukan. `TimeoutSeconds` Step Functions mencoba ulang

pemanggilan fungsi Lambda dalam status `Task` ini dua kali, dengan backoff eksponensial di antara percobaan ulang.

```
{
   "Comment": "A Hello World example of the Amazon States Language using an AWS Lambda
 function",
   "StartAt": "HelloWorld",
   "States": {
      "HelloWorld": {
         "Type": "Task",
         "Resource": "arn:aws:lambda:us-east-1:123456789012:function:sleep10",
         "TimeoutSeconds": 2,
         "Retry": [ {
            "ErrorEquals": ["States.Timeout"],
            "IntervalSeconds": 1,
            "MaxAttempts": 2,
            "BackoffRate": 2.0
         } ],
         "End": true
      }
   }
}
```

Menangani batas waktu menggunakan Catch

Contoh ini menggunakan bidang `Catch`. Ketika batas waktu terjadi, mesin status bertransisi ke status `fallback`.

```
{
   "Comment": "A Hello World example of the Amazon States Language using an AWS Lambda
 function",
   "StartAt": "HelloWorld",
   "States": {
      "HelloWorld": {
         "Type": "Task",
         "Resource": "arn:aws:lambda:us-east-1:123456789012:function:sleep10",
         "TimeoutSeconds": 2,
         "Catch": [ {
            "ErrorEquals": ["States.Timeout"],
            "Next": "fallback"
         } ],
         "End": true
      },
```

```
      "fallback": {
         "Type": "Pass",
         "Result": "Hello, AWS Step Functions!",
         "End": true
      }
   }
}
```

> **Note**
>
> Anda dapat mempertahankan input status dan kesalahan dengan menggunakan `ResultPath`. Lihat Gunakan ResultPath untuk Menyertakan Kesalahan dan Input dalam `Catch`.

Panggil AWS Step Functions dari layanan lain

Anda dapat mengonfigurasi beberapa layanan lain untuk memanggil mesin status. Berdasarkan mesin negarajenis alur kerja, Anda dapat memanggil mesin negara secara asinkron atau serempak. Untuk memanggil mesin negara secara serempak, gunakan`StartSyncExecution`Panggilan API atau integrasi Amazon API Gateway dengan Alur Kerja Ekspres. Dengan pemanggilan asinkron, Step Functions menghentikan eksekusi alur kerja sampai token tugas dikembalikan. Namun, menunggu token tugas memang membuat alur kerja sinkron.

Layanan yang dapat Anda konfigurasikan untuk menjalankan Step Functions meliputi:

- AWS Lambda, menggunakan panggilan `StartExecution`.
- Gerbang API Amazon
- AmazonEventBridge
- AWS CodePipeline
- AWS IoTAturan Mesin
- AWS Step Functions

Pemanggilan Step Functions diatur oleh kuota `StartExecution`. Untuk informasi selengkapnya, lihat:

- Kuota

Baca Konsistensi di Step Functions

Pembaruan mesin status di AWS Step Functions bersifat konsisten pada akhirnya. Semua panggilan `StartExecution` dalam beberapa detik akan menggunakan ketentuan dan `roleArn` (Nama sumber daya Amazon untuk IAM role) yang diperbarui. Eksekusi yang dimulai segera setelah memanggil `UpdateStateMachine` mungkin menggunakan ketentuan mesin status dan `roleArn` sebelumnya.

Untuk informasi selengkapnya, lihat yang berikut:

- `UpdateStateMachine` di Referensi API AWS Step Functions
- Memperbarui alur kerja diMemulai dengan AWS Step Functions.

Penandaan di Step Functions

AWS Step Functions mendukung penandaan mesin status (baik Standar dan Express) dan aktivitas. Hal ini dapat membantu Anda melacak dan mengelola biaya yang terkait dengan sumber daya Anda, serta memberikan keamanan yang lebih baik di kebijakan AWS Identity and Access Management (IAM) Anda. Sumber daya Step Functions penandaan mengizinkannya untuk dikelola oleh AWS Resource Groups. Untuk informasi lebih lanjut tetang Resource Groups, lihat Panduan Pengguna AWS Resource Groups.

Untuk otorisasi berbasis tag, sumber daya eksekusi mesin negara seperti yang ditunjukkan pada contoh berikut mewarisi tag yang terkait dengan mesin negara.

```
arn:<partition>:states:<Region>:<account-id>:execution:<StateMachineName>:<ExecutionId>
```

Saat Anda menelepon DescribeExecutionatau API lain di mana Anda menentukan ARN sumber daya eksekusi, Step Functions menggunakan tag yang terkait dengan state machine untuk menerima atau menolak permintaan saat melakukan otorisasi berbasis tag. Ini membantu Anda mengizinkan atau menolak akses ke eksekusi mesin negara di tingkat mesin negara.

Untuk meninjau pembatasan yang terkait dengan penandaan sumber daya, lihat Pembatasan terkait penandaan.

Topik

- Penandaan untuk Alokasi Biaya

- Penandaan untuk Keamanan
- Melihat dan Mengelola Tanda di Konsol Step Functions
- Kelola Tanda dengan Tindakan API Step Functions

Penandaan untuk Alokasi Biaya

Untuk mengatur dan mengidentifikasi sumber daya Step Functions Anda untuk alokasi biaya, Anda dapat menambahkan tanda metadata yang mengidentifikasi tujuan dari mesin status atau aktivitas. Ini sangat berguna jika Anda memiliki banyak sumber daya. Anda juga dapat menggunakan tanda untuk mengatur tagihan AWS Anda untuk merefleksikan struktur biaya Anda sendiri. Untuk melakukannya, daftar untuk membuat tagihan akun AWS Anda menyertakan kunci dan nilai tanda. Untuk informasi selengkapnya, lihat Menyiapkan Laporan Alokasi Biaya Bulanan di Panduan Pengguna AWS Billing.

Misalnya, Anda dapat menambahkan tanda yang mewakili pusat biaya dan tujuan sumber daya Step Functions Anda, sebagai berikut.

Resource	Kunci	Nilai
StateMachine1	Cost Center	34567
	Application	Image processing
StateMachine2	Cost Center	34567
	Application	Rekognition processing
Activity1	Cost Center	12345
	Application	Legacy database

Skema penandaan ini mengizinkan Anda untuk mengelompokkan dua mesin status melakukan tugas terkait di pusat biaya yang sama, ketika menandai aktivitas yang tidak terkait dengan tanda alokasi biaya yang berbeda.

Penandaan untuk Keamanan

IAM mendukung akses pengendalian ke sumber daya berdasarkan tanda. Untuk mengontrol akses berdasarkan tanda, Anda memberikan informasi tanda sumber daya di elemen syarat dari kebijakan IAM.

Misalnya, Anda dapat membatasi akses ke semua sumber daya Step Functions yang menyertakan tanda dengan kunci `environment` dan nilai `production`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "states:TagResource",
                "states:DeleteActivity",
                "states:DeleteStateMachine",
                "states:StopExecution"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/environment": "production"}
            }
        }
    ]
}
```

Untuk informasi selengkapnya, lihat Mengendalikan Akses Menggunakan Tanda di Panduan Pengguna IAM.

Melihat dan Mengelola Tanda di Konsol Step Functions

Step Functions mengizinkan Anda untuk melihat dan mengelola tanda untuk mesin status Anda di konsol Step Functions. Dari halaman Detail mesin status, pilih Tanda. Di sini, Anda dapat melihat tanda yang ada terkait dengan mesin status Anda.

> **Note**
>
> Untuk mengelola tanda untuk aktivitas, lihat Kelola Tanda dengan Tindakan API Step Functions.

Untuk menambah atau menghapus tanda yang terkait dengan mesin status Anda, pilih tombol Kelola Tanda.

1. Telusuri halaman detail mesin status.
2. Pilih Tanda, di samping Eksekusi dan Ketentuan.
3. Pilih Kelola tanda.
   - Untuk memodifikasi tanda yang ada, edit Kunci dan Nilai.
   - Untuk menghapus tanda yang ada, pilih Hapus tanda.
   - Untuk menambahkan tanda baru, pilih Tambahkan tanda dan edit Kunci dan Nilai.
4. Pilih Simpan.

Kelola Tanda dengan Tindakan API Step Functions

Untuk mengelola tanda menggunakan API Step Functions, gunakan tindakan API berikut:

- `ListTagsForResource`
- `TagResource`
- `UntagResource`

AWS Step Functions Studio Alur Kerja

Workflow Studio for AWS Step Functions adalah desainer alur kerja visual kode rendah yang memungkinkan Anda membuat alur kerja tanpa server dengan mengatur layanan. AWS Dengan menggunakan drag-and-drop fiturnya atau editor kode bawaan, Anda dapat membuat dan mengedit alur kerja, mengontrol bagaimana input dan output disaring atau diubah untuk setiap status, dan mengonfigurasi penanganan kesalahan. Saat Anda menyeret dan melepas status untuk membangun alur kerja Anda, Workflow Studio memvalidasi pekerjaan Anda dan membuat kode secara otomatis. Anda dapat meninjau kode yang dihasilkan atau memperbarui definisi mesin status dalam editor kode. Setelah selesai, Anda dapat menyimpan alur kerja, menjalankannya, lalu memeriksa hasilnya di konsol Step Functions. Anda dapat menambahkan dan memodifikasi alur kerja secara visual untuk mengatur beberapa layanan dalam aplikasi Anda.

Untuk menggunakan Step Functions Workflow Studio, Anda memerlukan Akun AWS, dan kredensional yang memberikan izin yang benar untuk sumber daya apa pun yang ingin Anda gunakan. Untuk informasi selengkapnya, lihat Prasyarat untuk Memulai AWS Step Functions.

> **Note**
>
> Workflow Studio tidak mendukung Internet Explorer 11. Jika Anda menggunakan Internet Explorer 11 dan mengalami masalah saat menggunakan Workflow Studio, coba gunakan browser lain.

Anda dapat mengakses Workflow Studio dari Konsol Step Functions, saat Anda membuat atau mengedit alur kerja di Step Functions. Anda juga dapat mengakses Workflow Studio di dalamnyaKomposer Aplikasi AWS. Workflow Studio in Application Composer menyediakan lingkungan IAc visual yang memudahkan Anda untuk menggabungkan alur kerja dalam aplikasi tanpa server yang dibangun menggunakan alat IAc, seperti template. AWS CloudFormation Menggunakan Workflow Studio inApplication Composer, Anda dapat membangun alur kerja menggunakan AWS CloudFormation template. Di dalamnyaApplication Composer, Anda dapat menambahkan alur kerja baru, memodifikasi alur kerja yang ada, dan menghubungkan langkah alur kerja individual ke sumber daya aplikasi lainnya. Application Composersecara otomatis membuat dan memperbarui CloudFormation sumber daya dan konfigurasi yang diperlukan. Ini membantu Anda membuat dan mengelola semua sumber daya yang digunakan dalam alur kerja Anda di satu tempat. Ini juga membantu Anda mempercepat jalur Anda dari pembuatan prototipe alur kerja hingga penerapan produksi.

Saat Anda menggunakan Workflow Studio diApplication Composer, Anda juga dapat langsung terhubung ke proyek lokal Anda. Ini membantu Anda bekerja di lingkungan pengembangan terintegrasi (IDE) Anda di samping kanvas visual. Untuk informasi selengkapnya, lihat Menggunakan Workflow Studio di Application Composer.

Topik

- Gambaran umum antarmuka
- Menggunakan Workflow Studio
- Konfigurasikan input dan output untuk status Anda
- Peran eksekusi di Workflow Studio
- Penanganan kesalahan
- Tutorial: Belajar untuk menggunakan Studio Alur Kerja AWS Step Functions

Gambaran umum antarmuka

Workflow Studio for AWS Step Functions adalah desainer alur kerja visual kode rendah yang memungkinkan Anda membuat alur kerja tanpa server dengan mengatur. Layanan AWS Dengan fitur drag and drop, Workflow Studio memudahkan Anda untuk membangun, mengedit, dan memvisualisasikan prototipe alur kerja Anda. Workflow Studio juga menawarkan editor kode bawaan untuk menulis dan mengedit definisi alur kerja Anda menggunakan Amazon States Language (ASL) dalam konsol Step Functions.

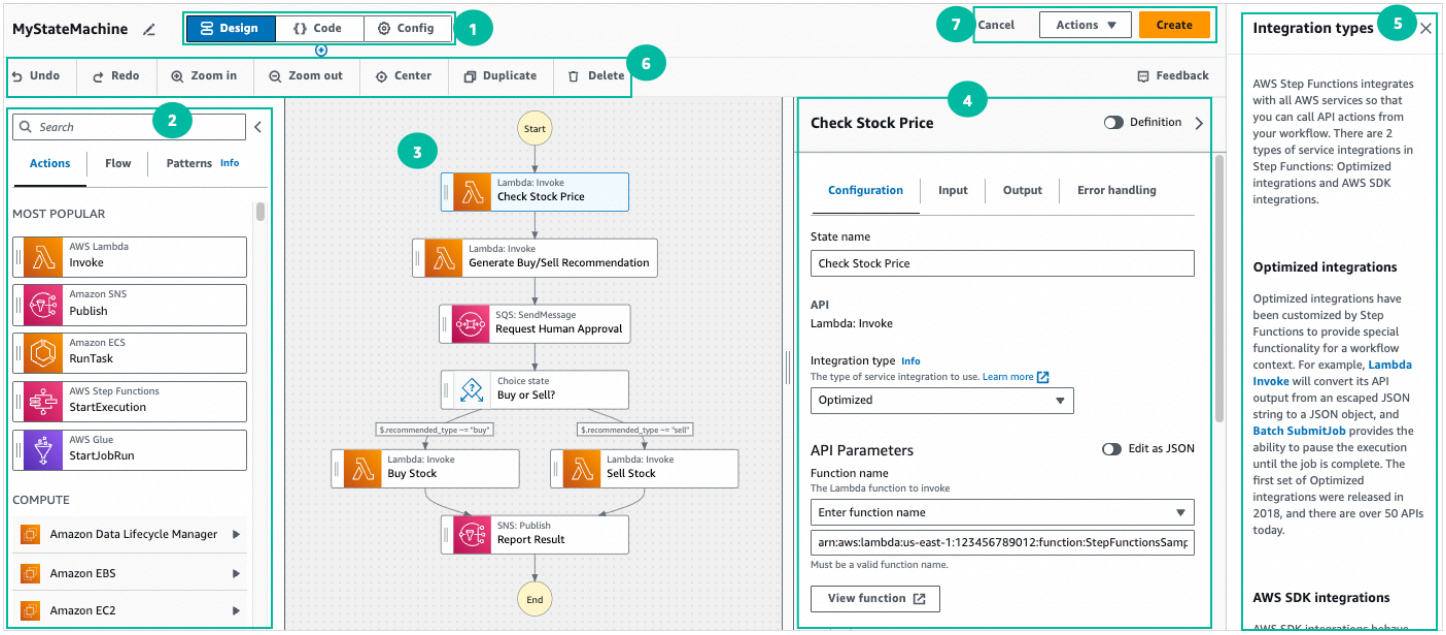
Untuk membantu Anda membangun dan memvisualisasikan alur kerja Anda, mengedit definisi mereka, dan mengelola konfigurasinya, Workflow Studio menyediakan tiga mode: Desain, Kode, dan Config. Bagian berikut menjelaskan mode ini secara rinci.

Dalam topik ini:

- Mode desain
- Mode kode
- Modus Config
- Pintasan keyboard

Mode desain

Mode Desain Workflow Studio menyediakan antarmuka grafis untuk memvisualisasikan alur kerja Anda saat Anda membangun prototipe mereka. Gambar berikut menunjukkan berbagai komponen yang tersedia dalam mode Desain.

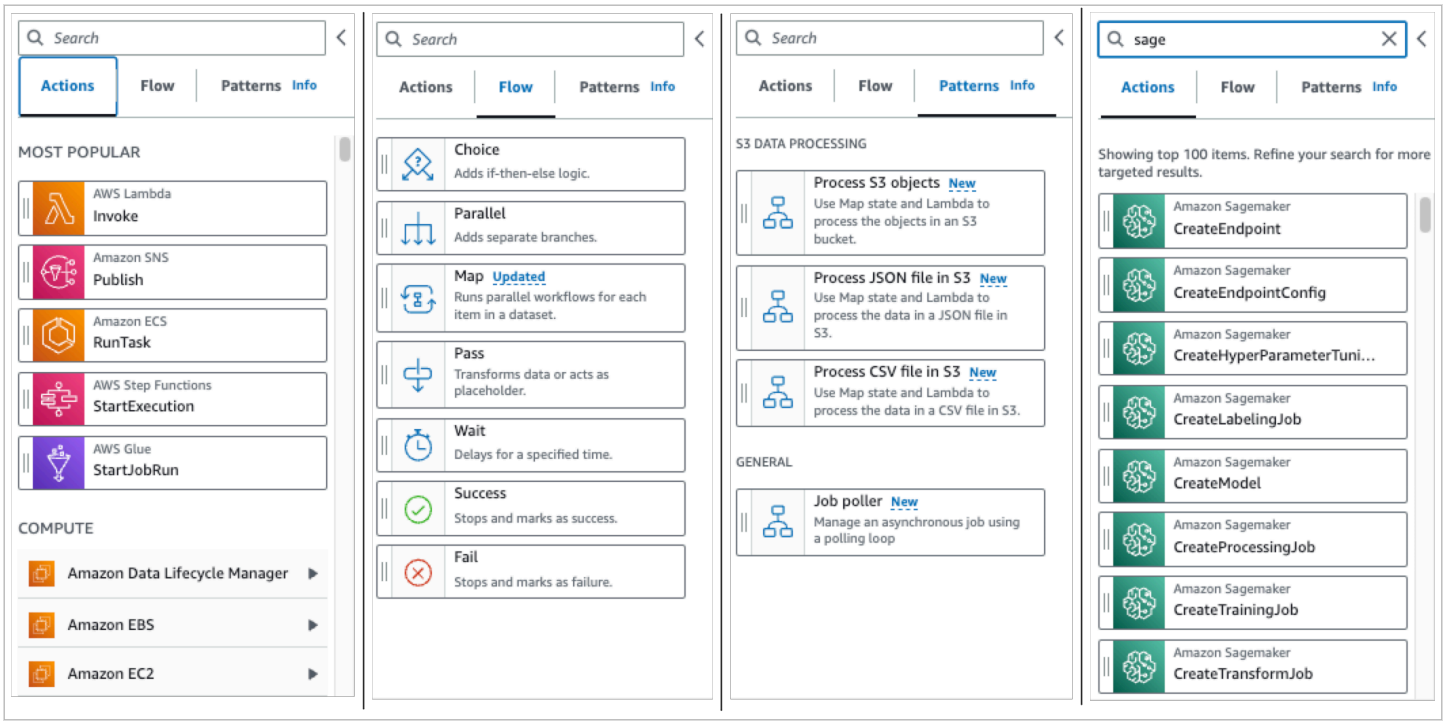


1. Tombol mode - Beralih antara mode Desain, Kode, dan Config dari Workflow Studio menggunakan tombol mode. Anda tidak dapat beralih mode jika JSON dalam definisi ASL alur kerja Anda tidak valid.
2. Peramban statusBerisi tiga tab berikut:
   - Tab Tindakan menyediakan daftar AWS API yang dapat Anda seret dan lepas ke grafik alur kerja di kanvas. Setiap tindakan mewakili suatu Tugas negara.
   - Tab Flow menyediakan daftar status alur yang dapat Anda seret dan lepas ke grafik alur kerja di kanvas.
   - Tab Patterns menyediakan beberapa ready-to-use blok bangunan yang dapat digunakan kembali yang dapat Anda gunakan untuk berbagai kasus penggunaan. Misalnya, Anda dapat menggunakan pola ini untuk memproses data secara berulang di bucket Amazon S3.
3. Di Kanvas sinilah Anda menyeret dan melepas status ke dalam grafik alur kerja Anda, mengubah urutan status, dan memilih status untuk dikonfigurasi atau dilihat.
4. InspectorPanel adalah tempat Anda dapat melihat dan mengedit properti dari setiap status yang Anda pilih di kanvas. Aktifkan sakelar Definisi untuk melihat kode Bahasa Negara Amazon untuk alur kerja Anda, dengan status yang dipilih saat ini disorot.

5. Tautan Info membuka panel dengan informasi kontekstual saat Anda memerlukan bantuan. Panel ini juga mencakup tautan ke topik terkait dalam dokumentasi Step Functions.
6. Toolbar desain - Berisi satu set tombol untuk melakukan tindakan umum, seperti membatalkan, menghapus, dan memperbesar.
7. Tombol utilitas — Satu set tombol untuk melakukan tugas, seperti menyimpan alur kerja Anda atau mengekspor definisi ASL mereka dalam file JSON atau YAMG.

Peramban status

Peramban Status adalah tempat Anda memilih status yang akan diseret dan dijatuhkan di grafik alur kerja Anda. Tab Tindakan menyediakan daftar AWS API, dan tab Flow menyediakan daftar status aliran. Sementara tab Patterns menyediakan beberapa ready-to-use blok bangunan yang dapat digunakan kembali yang dapat Anda gunakan untuk berbagai kasus penggunaan. Anda dapat mencari semua negara bagian di Browser Negara menggunakan kotak Pencarian di bagian atas.



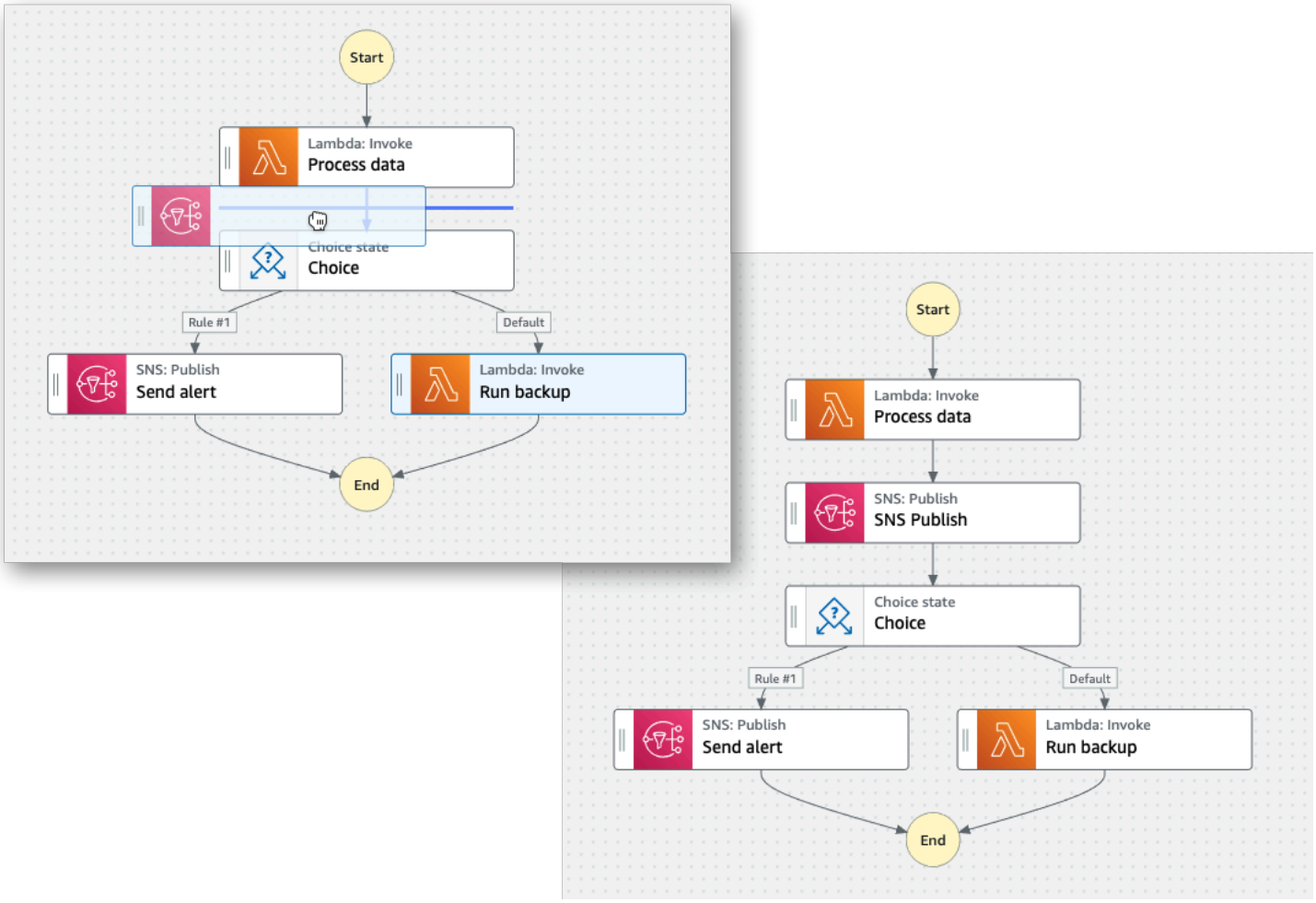
Ada tujuh status alur yang dapat Anda gunakan untuk mengarahkan dan mengontrol alur kerja Anda. Semua alur tersebut mengambil input dari status sebelumnya, dan sebagian besar memungkinkan Anda untuk memfilter input dari status sebelumnya, dan output ke status selanjutnya. Status alur adalah:

- Pilihan: Tambahkan pilihan antara cabang eksekusi alur kerja Anda. Di tab Konfigurasi Inspector, Anda dapat mengonfigurasi aturan untuk menentukan status untuk alur kerja akan bertransisi.
- Paralel: Tambahkan cabang eksekusi paralel alur kerja Anda.
- Map: Secara dinamis, ulangi langkah untuk setiap elemen array input. Tidak seperti status alur `Parallel`, status `Map` akan mengeksekusi langkah yang sama untuk beberapa entri array di input status.
- Diteruskan: Memungkinkan Anda meneruskan inputnya ke outputnya. (Opsional) Anda dapat menambahkan data tetap ke output.
- Tunggu: Beri jeda alur kerja Anda selama waktu tertentu atau sampai waktu atau tanggal yang ditentukan.
- Berhasil: Hentikan alur kerja Anda dengan sukses.
- Gagal: Hentikan alur kerja Anda dengan gagal.

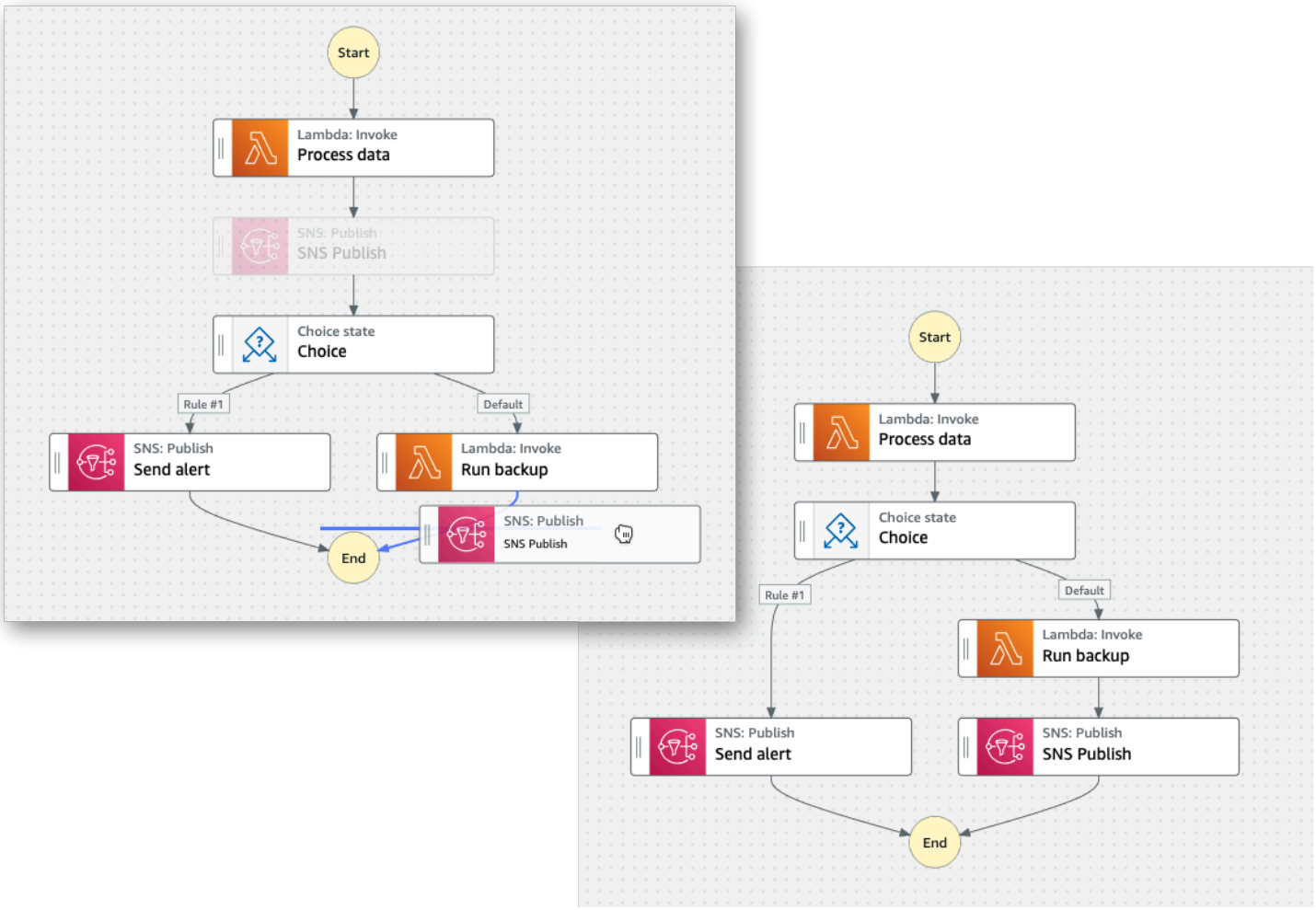
Kanvas

Setelah Anda memilih status yang akan ditambahkan ke alur kerja, seret status ke kanvas dan jatuhkan ke grafik alur kerja Anda. Anda juga dapat menyeret dan menjatuhkan status untuk memindahkannya ke tempat yang berbeda dalam alur kerja Anda. Jika alur kerja Anda rumit, Anda mungkin tidak dapat melihat semua status di panel kanvas. Gunakan kendali di bagian atas kanvas untuk memperbesar atau memperkecil. Untuk melihat bagian grafik alur kerja yang berbeda, Anda dapat menyeret grafik alur kerja di kanvas.

Seret status alur kerja dari tab Actions atau Flow dan jatuhkan ke dalam alur kerja Anda. Baris menunjukkan tempat status akan ditempatkan di alur kerja Anda. Status alur kerja baru telah ditambahkan ke alur kerja Anda, dan kodenya dibuat secara otomatis.

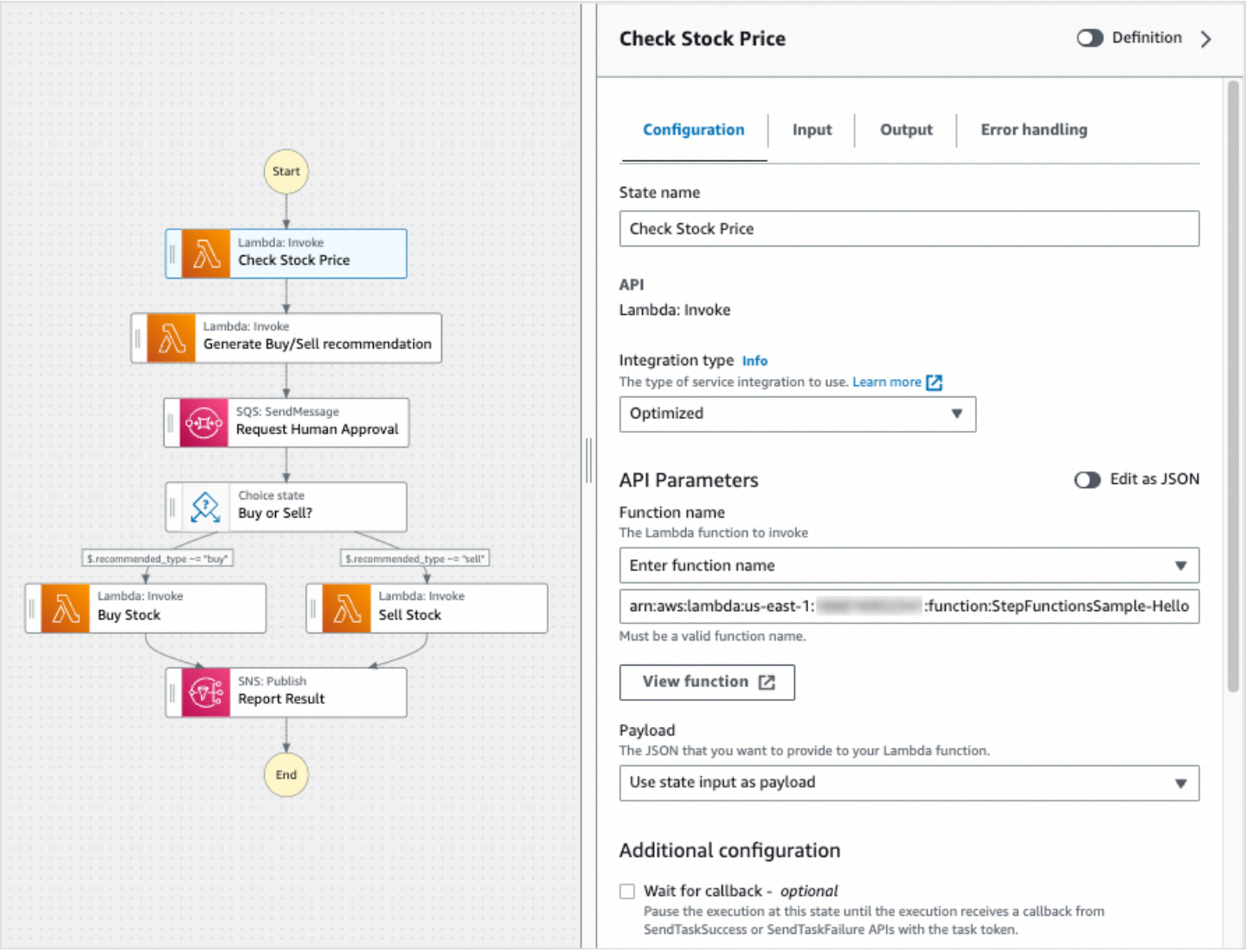


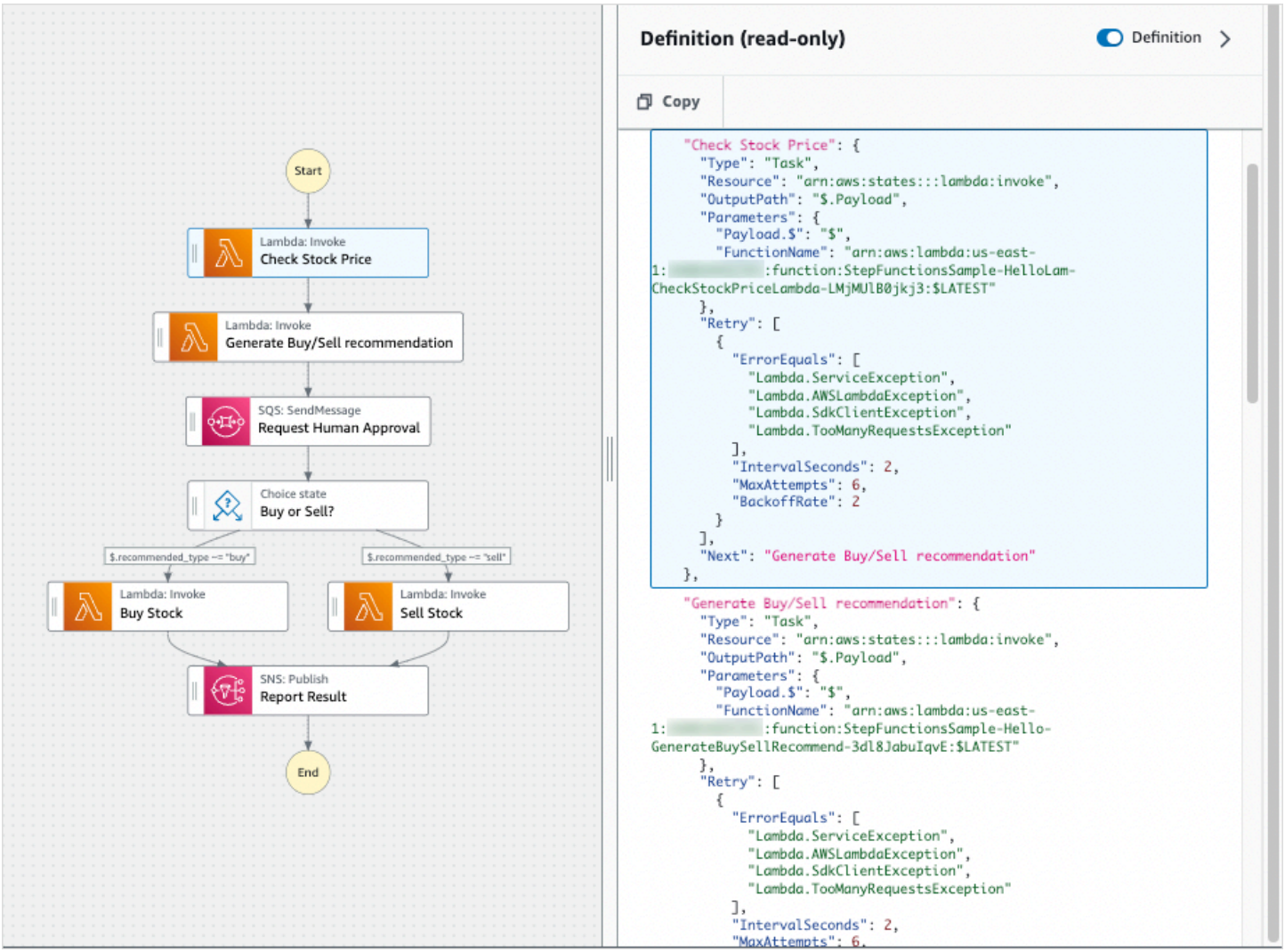
Untuk mengubah urutan status, Anda dapat menyeretnya ke tempat lain di alur kerja Anda.



Inspector

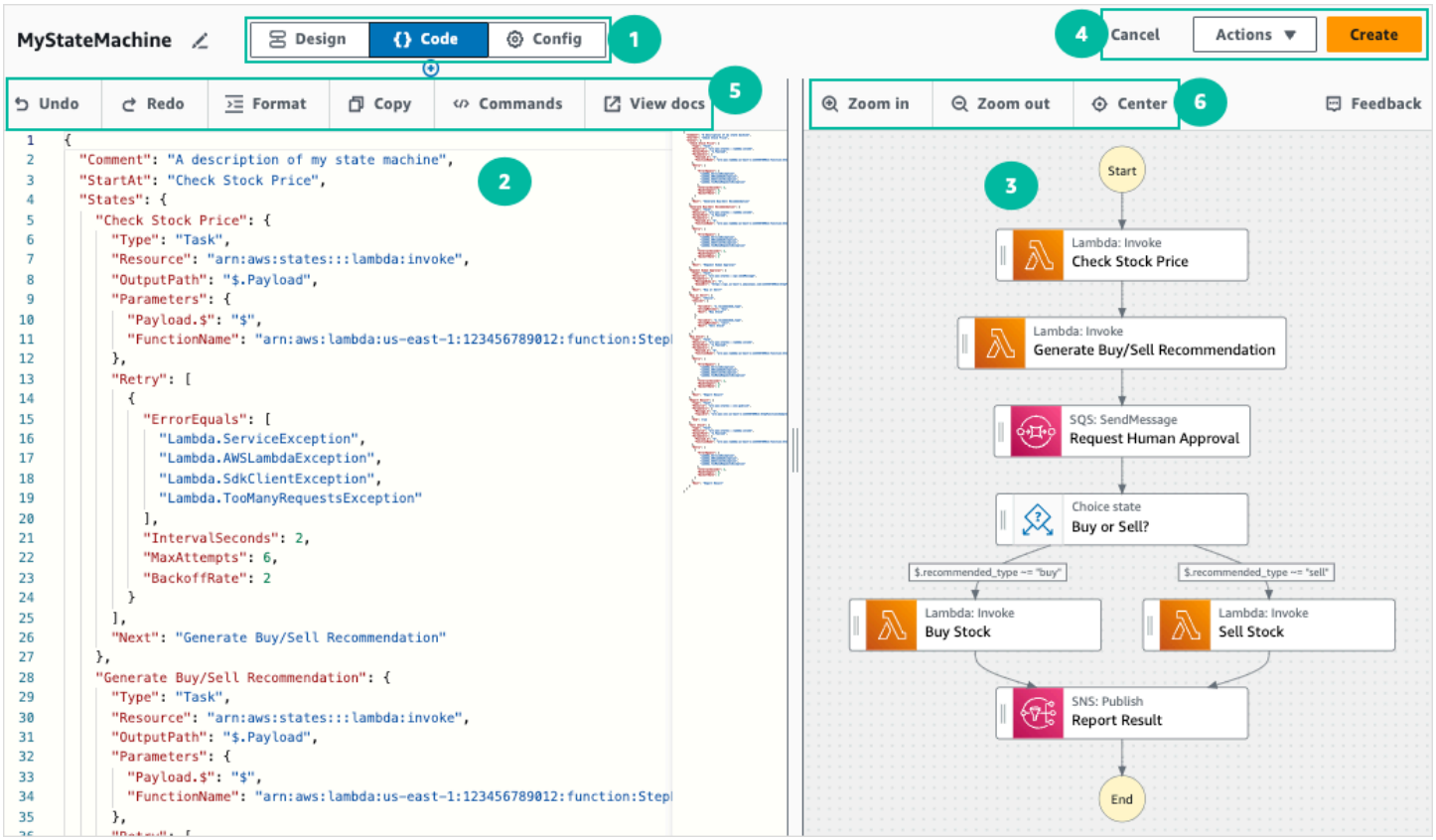
Anda dapat mengonfigurasi status apa pun yang Anda tambahkan ke alur kerja Anda. Pilih status yang ingin Anda konfigurasikan, dan Anda akan melihat opsi konfigurasinya di panel Inspector. Untuk melihat definisi ASL yang dibuat secara otomatis untuk kode alur kerja Anda, aktifkan sakelar Definisi. Definisi ASL yang terkait dengan status yang Anda pilih akan muncul disorot.


Start
Lambda: Invoke
Check Stock Price
Lambda: Invoke
Generate Buy/Sell recommendation
SQS: SendMessage
Request Human Approval
Choice state
Buy or Sell?
$.recommended_type == "buy"
$.recommended_type == "sell"
Lambda: Invoke
Buy Stock
Lambda: Invoke
Sell Stock
SNS: Publish
Report Result
End
Check Stock Price
Definition
Configuration
Input
Output
Error handling
State name
Check Stock Price
API
Lambda: Invoke
Integration type Info
The type of service integration to use. Learn more
Optimized
API Parameters
Edit as JSON
Function name
The Lambda function to invoke
Enter function name
arn:aws:lambda:us-east-1:
:function:StepFunctionsSample-Hello
Must be a valid function name.
View function
Payload
The JSON that you want to provide to your Lambda function.
Use state input as payload
Additional configuration
Wait for callback - optional
Pause the execution at this state until the execution receives a callback from SendTaskSuccess or SendTaskFailure APIs with the task token.



Mode kode

Mode Kode Workflow Studio menyediakan editor kode terintegrasi untuk melihat, menulis, dan mengedit definisi Amazon States Language (ASL) alur kerja Anda dalam konsol Step Functions. Gambar berikut menunjukkan berbagai komponen yang tersedia dalam mode Kode.



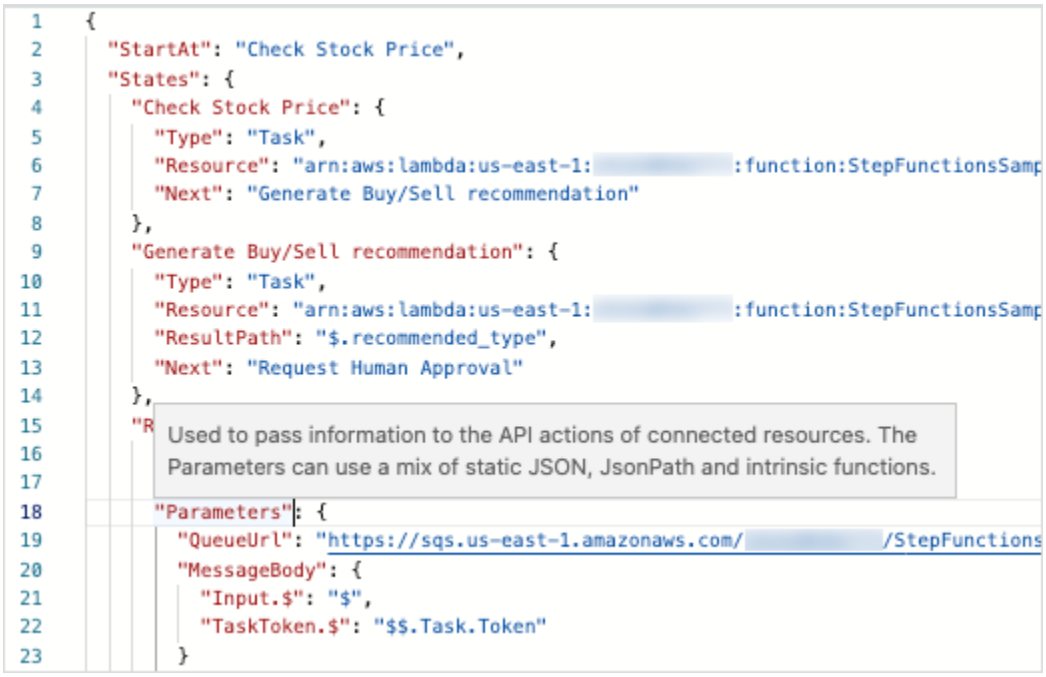
1. Tombol mode - Beralih antara mode Desain, Kode, dan Config dari Workflow Studio menggunakan tombol mode. Anda tidak dapat beralih mode jika JSON dalam definisi ASL alur kerja Anda tidak valid.
2. Di Editor kode sinilah Anda menulis dan mengedit definisi ASL dari alur kerja Anda dalam Workflow Studio. Editor kode juga menyediakan fitur, seperti penyorotan sintaks dan pelengkapan otomatis.
3. Panel visualisasi grafik— Menampilkan visualisasi grafis real-time dari alur kerja Anda.
4. Tombol utilitas — Satu set tombol untuk melakukan tugas, seperti menyimpan alur kerja Anda atau mengekspor definisi ASL mereka dalam file JSON atau YAMG.
5. Code toolbar - Berisi satu set tombol untuk melakukan tindakan umum, seperti membatalkan tindakan atau memformat kode.
6. Bilah alat grafik - Berisi sekumpulan tombol untuk melakukan tindakan umum, seperti memperbesar dan memperkecil grafik alur kerja.

Editor kode

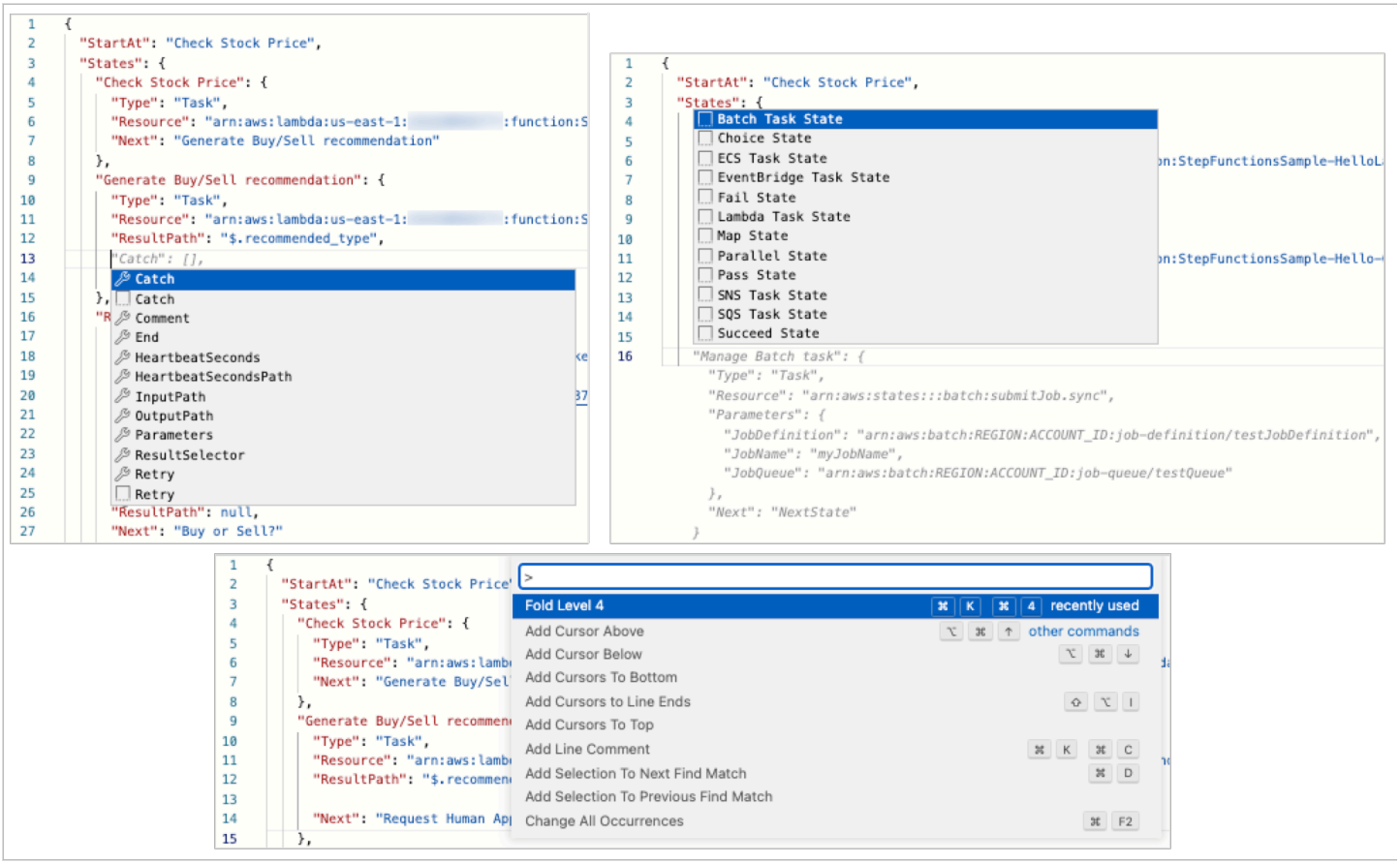
Editor kode memberikan pengalaman seperti IDE untuk menulis dan mengedit definisi alur kerja Anda menggunakan JSON dalam Workflow Studio. Editor kode mencakup beberapa fitur, seperti penyorotan sintaks, saran pelengkapan otomatis, validasi definisi ASL, dan tampilan bantuan peka konteks. Saat Anda memperbarui definisi alur kerja Anda, akan Panel visualisasi grafik membuat grafik real-time alur kerja Anda. Anda juga dapat melihat grafik alur kerja yang diperbarui di. Mode desain

Jika Anda memilih status di Mode desain atau panel visualisasi grafik, definisi ASL dari status tersebut akan disorot di editor kode. Definisi ASL alur kerja Anda diperbarui secara otomatis jika Anda menyusun ulang, menghapus, atau menambahkan status dalam mode Desain atau panel visualisasi grafik.

Tempatkan kursor di atas bidang apa pun dalam definisi alur kerja untuk melihat bantuan peka konteks sebagai tooltip.



Saran pelengkapan otomatis menampilkan cuplikan kode untuk bidang atau status yang dapat Anda sertakan dalam alur kerja Anda. Untuk melihat daftar bidang yang dapat Anda sertakan dalam status tertentu, tekan**`Ctrl+Space`**. Untuk menghasilkan cuplikan kode untuk status baru dalam alur kerja Anda, tekan **`Ctrl+Space`** setelah definisi status saat ini. Anda juga dapat menekan **`F1`** untuk menampilkan daftar perintah yang tersedia.



Panel visualisasi grafik

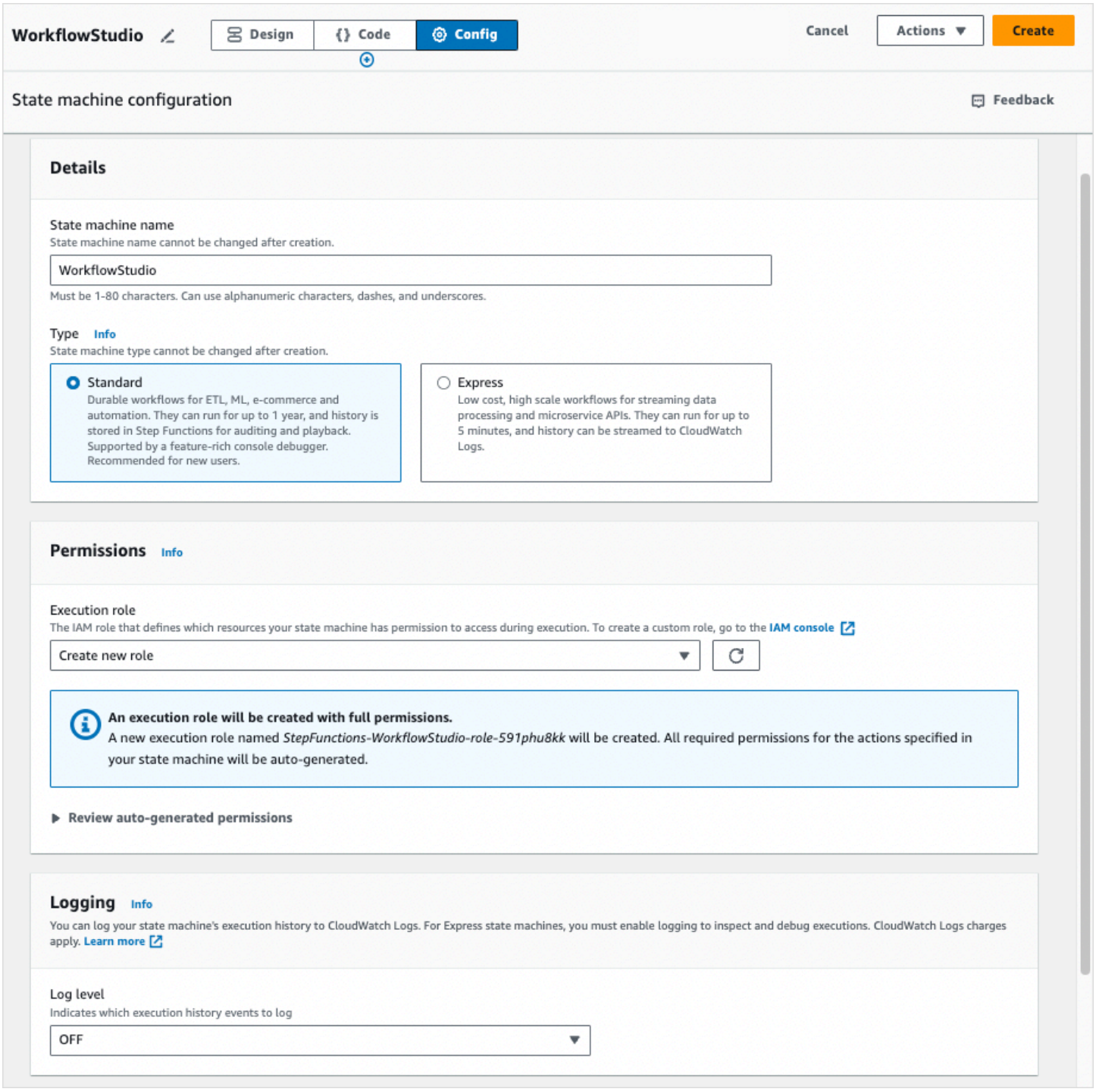
Visualisasi grafik memungkinkan Anda melihat seperti apa alur kerja Anda dalam format grafis. Saat Anda menulis definisi alur kerja Anda di Editor kode Workflow Studio, panel visualisasi grafik akan membuat grafik real-time alur kerja Anda. Saat Anda menyusun ulang, menghapus, atau menduplikasi status di panel visualisasi grafik, definisi alur kerja di editor Kode diperbarui secara otomatis. Demikian pula, saat Anda memperbarui definisi alur kerja Anda, menyusun ulang, menghapus, atau menambahkan status di editor Kode, visualisasi diperbarui secara otomatis.

Jika JSON dalam definisi ASL alur kerja Anda tidak valid, panel visualisasi grafik menjeda rendering dan menampilkan pesan status di bagian bawah panel.

Modus Config

Mode Config Workflow Studio memungkinkan Anda mengelola konfigurasi mesin status Anda. Dalam mode ini, Anda dapat menentukan detail, seperti nama mesin status dan jenisnya, izin IAM, dan konfigurasi logging untuk mesin status. Konfigurasi tambahan lainnya yang dapat Anda tentukan dalam mode ini termasuk mengaktifkan AWS X-Ray penelusuran dan penerbitan versi saat

Anda membuat mesin status. Setelah Anda membuat mesin status, Anda dapat mengedit semua opsi konfigurasi mesin status kecuali nama dan jenis mesin status. Gambar berikut menunjukkan beberapa konfigurasi yang dapat Anda tentukan dalam mode Config.



Kelola konfigurasi mesin status

Untuk mengelola konfigurasi state machine Anda, lakukan hal berikut:

1. Masukkan nama untuk mesin negara Anda di kotak Nama mesin Negara.

> **Tip**
>
> Atau, pilih ikon edit di sebelah nama mesin status default MyStateMachine. Kemudian, di bawah konfigurasi mesin negara, tentukan nama.

> **Important**
>
> Anda tidak dapat mengedit nama mesin status setelah Anda membuat mesin status.

2. Dalam Tipe, pilih jenis mesin status Standar atau Ekspres. Untuk informasi tentang jenis mesin negara bagian, lihatAlur Kerja Standar vs Ekspres.

> **Important**
>
> Anda tidak dapat mengedit jenis mesin status setelah Anda membuat mesin status.

3. Dalam Izin, pilih peran IAM yang akan digunakan sebagai peran eksekusi untuk mesin status.
   - Buat peran baru (Disarankan): Jika Anda memilih opsi ini, Step Functions secara otomatis membuat peran eksekusi untuk mesin status Anda dengan hak istimewa paling sedikit yang diperlukan saat Anda membuat mesin status. Peran IAM yang dihasilkan secara otomatis ini berlaku untuk Wilayah AWS di mana Anda membuat mesin status.

> **Tip**
>
> Untuk meninjau izin yang akan dihasilkan Step Functions secara otomatis untuk mesin status Anda, pilih Tinjau izin yang dibuat secara otomatis.

> **Note**
>
> Jika Anda menghapus IAM role yang Step Functions buat, Step Functions tidak dapat membuatnya kembali nanti. Demikian pula, jika Anda mengubah peran (misalnya, dengan menghapus Step Functions dari principal dalam kebijakan IAM), Step Functions tidak dapat memulihkan pengaturan aslinya nanti.

- Pilih peran yang ada: Buat peran IAM Anda sendiri untuk mesin status dan kemudian pilih dari opsi yang tercantum di bawah ini Pilih peran yang ada. Pastikan kebijakan peran menyertakan izin yang Anda ingin diasumsikan oleh mesin status.

  Untuk informasi tentang pembuatan kebijakan IAM, lihat Membuat kebijakan IAM di Panduan Pengguna IAM.

- Masukkan peran ARN: Tentukan Nama Sumber Daya Amazon (ARN) dari peran IAM yang ada untuk digunakan untuk mesin status ini. Misalnya, `arn:aws:iam::123456789012:role/service-role/StepFunctions-WorkflowStudio-role-777f4027`.

4. Di Logging, atur level log untuk mesin status Anda. Step Functions mencatat peristiwa riwayat eksekusi berdasarkan pilihan Anda. Anda dapat memilih salah satu opsi berikut:
   - SEMUA: Semua jenis acara dicatat.
   - ERROR: Semua jenis peristiwa kesalahan dicatat, seperti TaskFailed dan ExecutionFailed.
   - FATAL: Semua jenis peristiwa kesalahan fatal dicatat, seperti ExecutionAborted dan ExecutionFailed.
   - OFF: Tidak ada jenis acara yang dicatat.

   Untuk informasi selengkapnya tentang tingkat log, lihatTingkat Log.
5. Dalam konfigurasi Tambahan, atur satu atau beberapa konfigurasi opsional berikut:
   - Aktifkan X-Ray penelusuran: Pilih kotak centang ini untuk Step Functions X-Ray untuk mengirim jejak untuk eksekusi mesin status, bahkan ketika ID pelacakan tidak diteruskan oleh layanan upstream. Untuk informasi selengkapnya, lihat AWS X-Ray dan Step Functions.
   - Publikasikan versi saat pembuatan: Versi adalah snapshot bernomor dan tidak dapat diubah dari mesin status yang dapat Anda jalankan. Pilih kotak centang ini untuk mempublikasikan versi mesin status Anda saat membuat mesin status. Step Functions menerbitkan versi 1 sebagai revisi pertama mesin negara.

     Untuk informasi selengkapnya tentang versi, lihatVersi mesin negara.
   - Tambahkan tag baru: Pilih kotak ini untuk menambahkan tag ke mesin status Anda. Menambahkan tag dapat membantu Anda melacak dan mengelola biaya yang terkait dengan sumber daya Anda, dan memberikan keamanan yang lebih baik dalam kebijakan IAM Anda. Untuk informasi selengkapnya tentang tag, lihat Penandaan di Step Functions.
6. Pilih Buat.
7. Dalam kotak dialog Konfirmasi pembuatan peran, pilih Konfirmasi untuk melanjutkan.

Anda juga dapat memilih Lihat konfigurasi peran untuk kembali ke mode Config.

Pintasan keyboard

Workflow Studio mendukung pintasan keyboard berikut:

Pintasan keyboard	Fungsi
Shortcuts for the Code mode	
Ctrl+space	Auto-complete suggestions
F1	Display a list of available commands
Common shortcuts for the Design and Code modes	
Ctrl+Z	Undo the last operation
Ctrl+Shift+Z	Redo the last operation
Alt+C	Center the workflow in the canvas
Backspace	Remove all selected states
Delete	Remove all selected states
Ctrl+D	Duplicate selected state

Menggunakan Workflow Studio

Pelajari cara membuat, mengedit, dan menjalankan alur kerja menggunakan Step Functions Workflow Studio. Setelah alur kerja siap, Anda dapat mengekspornya. Anda juga dapat menggunakan Workflow Studio untuk melakukan prototipe cepat.

Dalam topik ini:

- Buat alur kerja
- Rancang alur kerja
- Jalankan alur kerja Anda

- Edit alur kerja
- Ekspor alur kerja Anda
- Buat prototipe alur kerja Anda

Buat alur kerja

Di Workflow Studio, Anda dapat memilih template pemula, atau memilih template kosong untuk membuat alur kerja dari awal. Untuk template kosong, Anda dapat menggunakan mode Desain atau Kode untuk membuat alur kerja Anda.

Template pemula adalah proyek ready-to-run contoh yang secara otomatis membuat proptotype dan definisi alur kerja, dan menyebarkan semua AWS sumber daya terkait yang dibutuhkan proyek Anda ke proyek Anda. Akun AWS Anda dapat menggunakan template pemula ini untuk menyebarkan dan menjalankannya apa adanya, atau menggunakan prototipe alur kerja untuk membangunnya. Untuk informasi selengkapnya tentang template pemula, lihatProyek sampel untuk Step Functions.

Buat alur kerja menggunakan template pemula

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Dalam kotak dialog Pilih templat, lakukan salah satu hal berikut untuk memilih proyek sampel, misalnya, proyek sampel Pengatur Waktu Tugas:

   - Ketik **`Task Timer`** kotak Cari menurut kata kunci, lalu pilih Pengatur Waktu Tugas dari hasil pencarian yang dikembalikan.
   - Jelajahi proyek sampel yang tercantum di bawah Semua di panel kanan, lalu pilih Pengatur Waktu Tugas.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.
5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang

tercantum dalam definisi alur kerja. DiMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke definisi Mode kode untuk memperbarui Amazon States Language (ASL) alur kerja Anda.

> **Important**
>
> Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

  > **Tip**
  >
  > Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

  Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

  > **Note**
  >
  > Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

  > **Important**
  >
  > Biaya standar berlaku untuk setiap layanan yang digunakan dalam CloudFormation template.

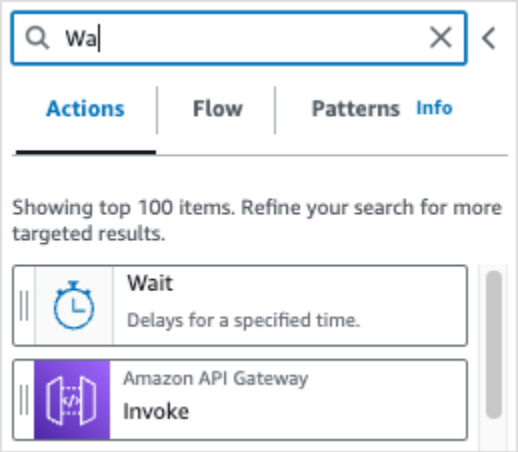
Buat alur kerja menggunakan template kosong

1. Buka Konsol Step Functions.
2. Pilih Buat mesin status.
3. Dalam kotak dialog Pilih templat, pilih Kosong.
4. Pilih Pilih. Ini membuka Workflow Studio diMode desain.

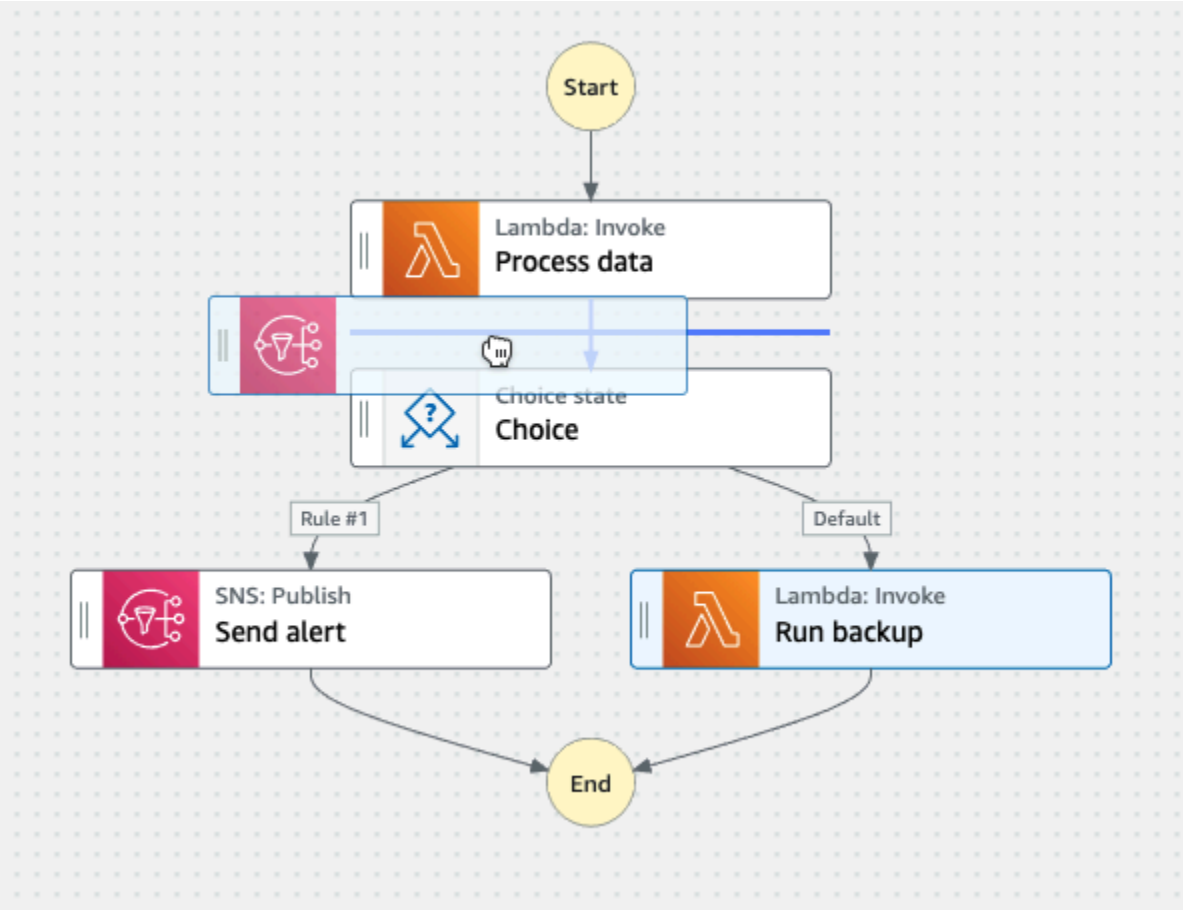
   Anda sekarang dapat mulai merancang alur kerja Anda Mode desain atau menulis definisi alur kerja Anda di. Mode kode
5. Pilih Config untuk mengelola konfigurasi alur kerja Anda di. Modus Config Misalnya, berikan nama untuk alur kerja Anda dan pilih jenisnya.

Rancang alur kerja

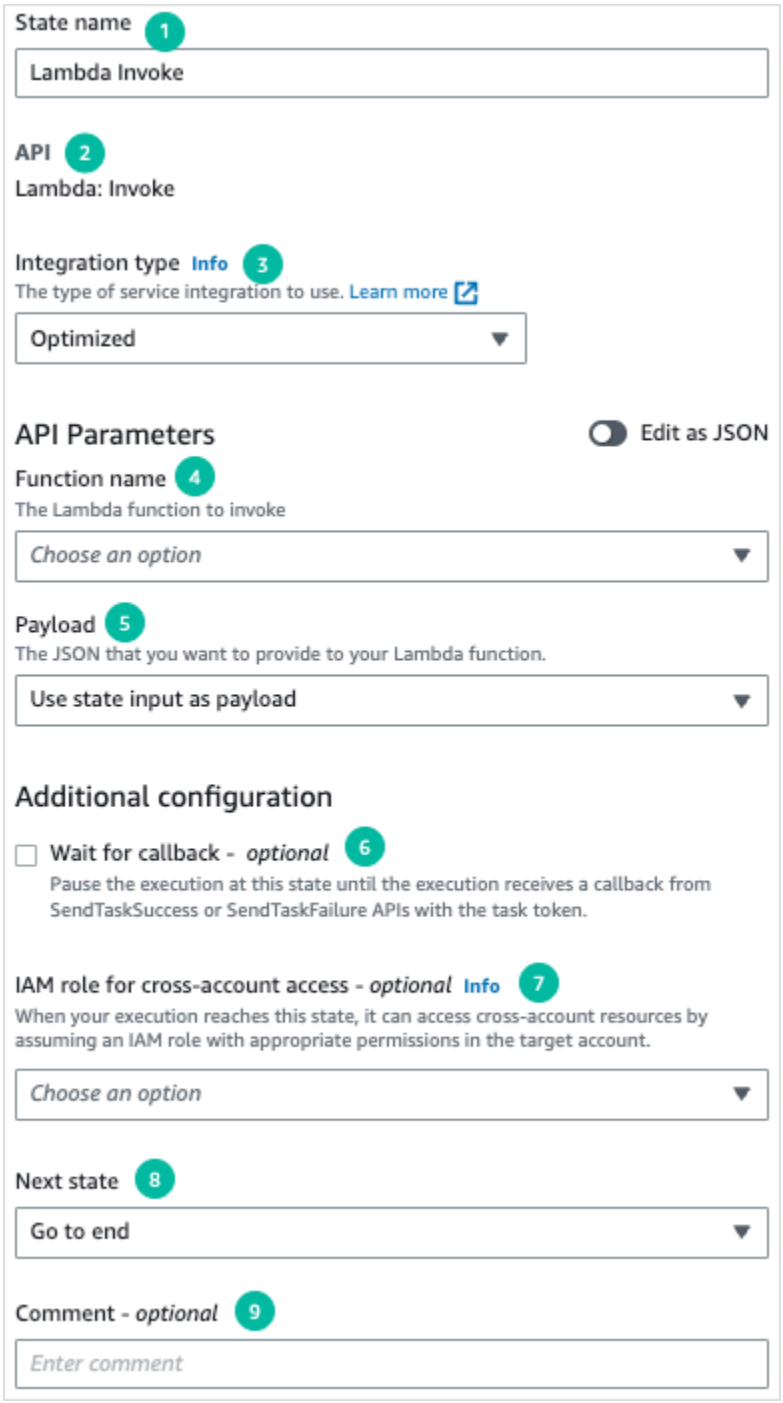
Jika Anda tahu nama negara yang ingin Anda tambahkan, gunakan kotak pencarian di bagian atas Peramban status untuk menemukan status tersebut di tab Tindakan dan Alur. Mode desain



Jika tidak, pilih status dari browser status dan seret dan jatuhkan ke kanvas, letakkan di tempat yang Anda inginkan dalam alur kerja Anda. Anda juga dapat menyusun ulang status dalam alur kerja Anda dengan menyeretnya ke lokasi yang berbeda dalam alur kerja Anda. Saat Anda menyeret status ke kanvas, garis akan muncul di tempat mana pun Anda dapat menjatuhkannya di alur kerja Anda. Setelah status dijatuhkan ke kanvas, kodenya dibuat secara otomatis dan ditambahkan di dalam definisi alur kerja Anda. Untuk melihat definisi, aktifkan sakelar Definisi pada panel Inspector. Untuk mengedit definisi alur kerja Anda, pilih Mode kode yang menawarkan editor kode terintegrasi.



Setelah Anda menjatuhkan status ke kanvas, Anda dapat mengonfigurasinya di Inspector panel di sebelah kanan. Panel ini berisi tab Konfigurasi, Input, Output, dan Penanganan Kesalahan untuk setiap status atau tindakan API yang Anda tempatkan di kanvas. Anda mengonfigurasi status yang Anda sertakan dalam alur kerja Anda di tab Konfigurasi. Misalnya, tab Konfigurasi untuk tindakan API Lambda Invoke terdiri dari opsi berikut:

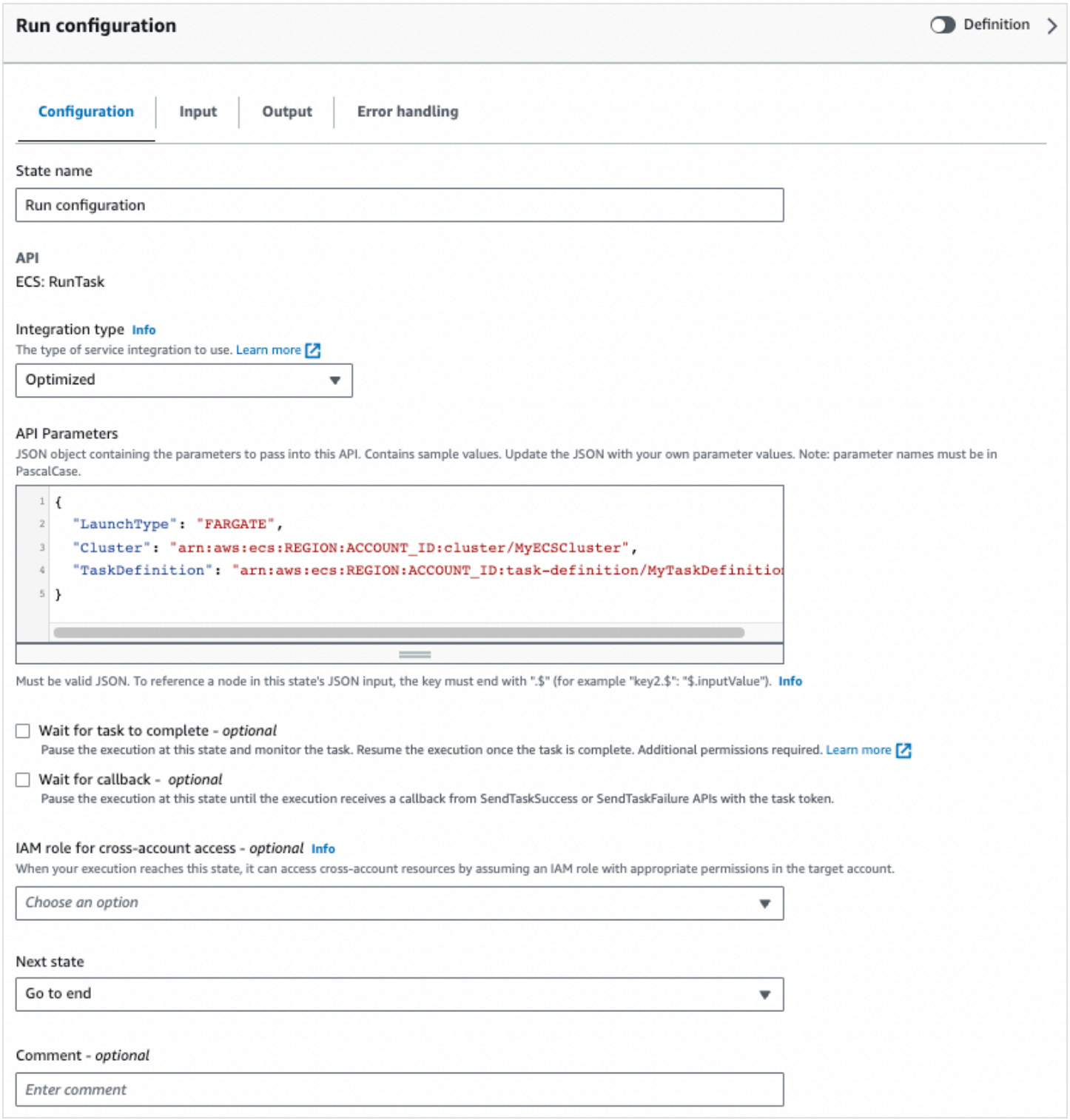


1. Nama status mengidentifikasi status. Anda dapat menggunakan nama Anda sendiri atau menerima nama default yang dibuat.
2. API menunjukkan tindakan API yang digunakan oleh status.
3. Daftar tarik-turun tipe Integrasi menyediakan opsi untuk memilih jenis integrasi layanan yang tersedia di Step Functions. Jenis integrasi yang Anda pilih digunakan untuk memanggil tindakan API tertentu Layanan AWS dari alur kerja Anda.

4. Nama fungsi menyediakan opsi untuk:
   - Masukkan nama fungsi: Anda dapat memasukkan nama fungsi Anda atau ARN nya.
   - Dapatkan nama fungsi pada saat waktu aktif input status: Anda dapat menggunakan opsi ini untuk secara dinamis mendapatkan nama fungsi dari input status berdasarkan jalur yang Anda tentukan.
   - Pilih nama fungsi: Anda dapat langsung memilih dari fungsi yang tersedia di akun dan wilayah Anda.
5. Payload memungkinkan Anda memilih dari opsi berikut:
   - Gunakan input status sebagai payload: Anda dapat menggunakan opsi ini untuk meneruskan masukan status sebagai payload yang diberikan ke fungsi Lambda Anda.
   - Masukkan payload Anda sendiri: Anda dapat menggunakan opsi ini untuk membangun objek JSON untuk diteruskan sebagai payload ke fungsi Lambda Anda. JSON ini dapat mencakup nilai statis dan nilai-nilai yang dipilih dari input status.
   - Tanpa muatan: Anda dapat menggunakan opsi ini jika Anda tidak ingin meneruskan muatan apa pun ke fungsi Lambda Anda.
6. (Opsional) Beberapa negara akan memiliki opsi untuk memilih Tunggu tugas selesai atau Tunggu panggilan balik. Saat tersedia, opsi ini memilih salah satu pola integrasi layanan dari berikut ini:
   - Tidak ada opsi yang dipilih: Step Functions akan menggunakan pola integrasi Minta Tanggapan. Step Functions akan menunggu respons HTTP lalu melanjutkan ke status berikutnya. Step Functions tidak akan menunggu tugas selesai. Bila tidak ada pilihan yang tersedia, negara akan menggunakan pola ini.
   - Tunggu tugas selesai: Step Functions akan menggunakan pola integrasi Jalankan Tugas (.sync).
   - Tunggu panggilan balik: Step Functions akan menggunakan pola integrasi Tunggu Panggilan Balik dengan Token Tugas.
7. (Opsional) Untuk mengakses sumber daya yang dikonfigurasi berbeda Akun AWS dalam alur kerja Anda, Step Functions menyediakan akses lintas akun. Peran IAM untuk akses lintas akun menyediakan opsi untuk:
   - Berikan ARN peran IAM: Tentukan peran IAM yang berisi izin akses sumber daya yang sesuai. Sumber daya ini tersedia di akun target, yang merupakan Akun AWS tempat Anda melakukan panggilan lintas akun.
   - Dapatkan ARN peran IAM saat runtime dari input status: Tentukan jalur referensi ke pasangan nilai kunci yang ada di input JSON status yang berisi peran IAM.

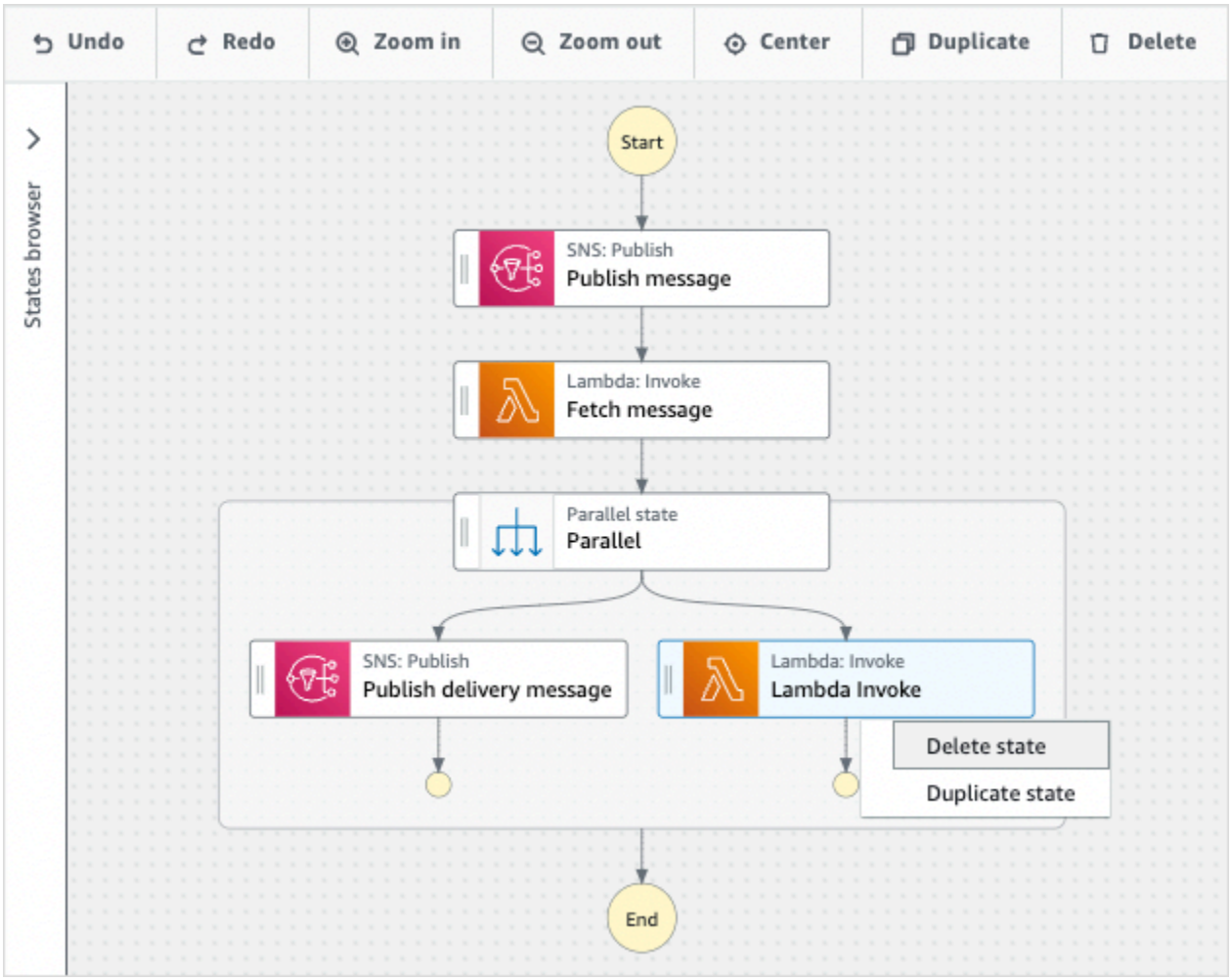
8. Status selanjutnya memungkinkan Anda untuk memilih status yang ingin Anda transisikan ke berikutnya.
9. (Opsional) Bidang Komentar dapat digunakan untuk menambahkan komentar Anda sendiri. Ini tidak akan memengaruhi alur kerja, tetapi dapat digunakan untuk membuat anotasi alur kerja Anda.

Beberapa status akan memiliki opsi konfigurasi yang lebih umum. Misalnya, konfigurasi status `RunTask` Amazon ECS berisi bidang `API Parameters` yang diisi dengan nilai placeholder.



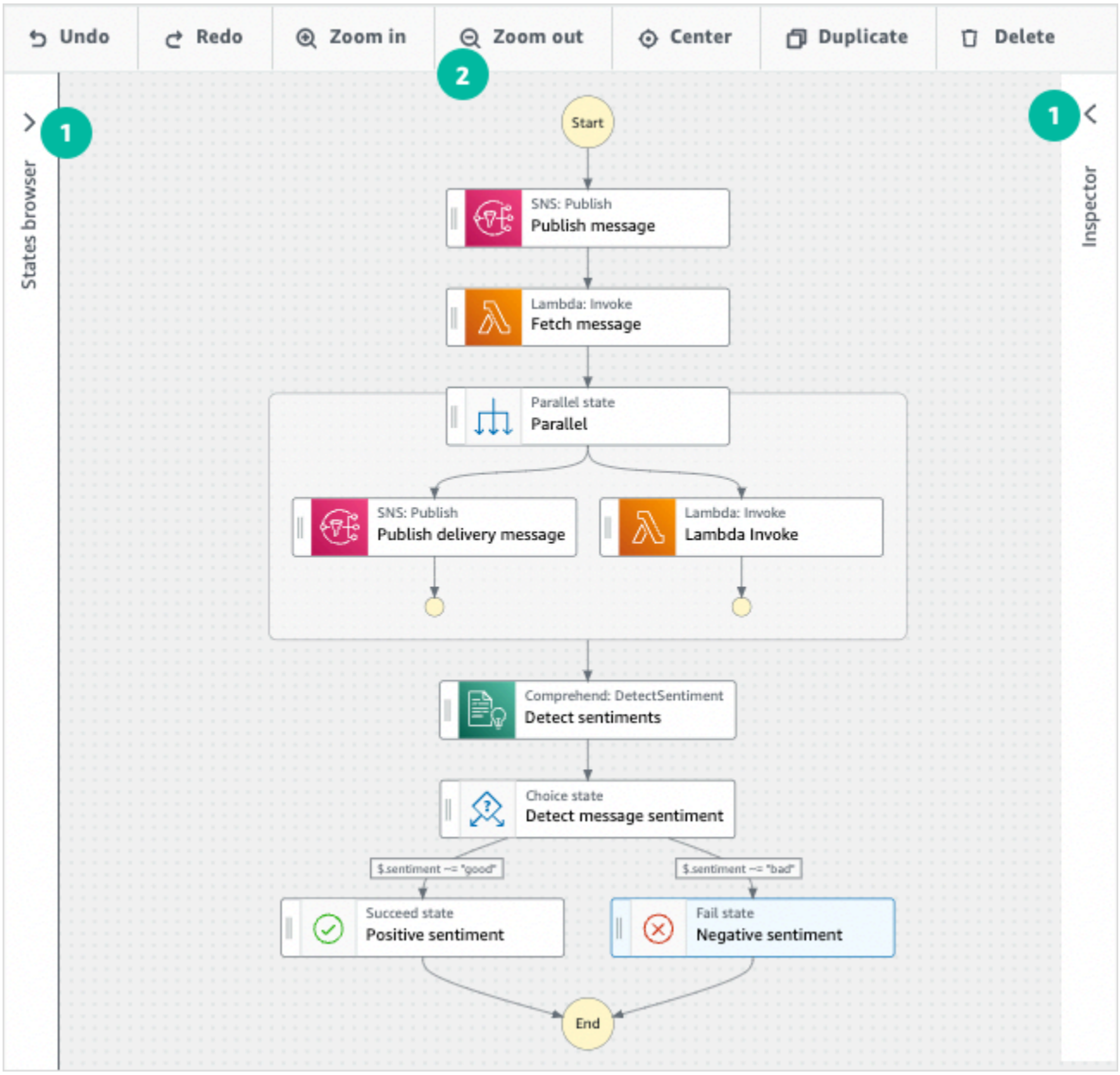
Untuk status ini, Anda dapat mengganti nilai placeholder dengan konfigurasi yang sesuai dengan kebutuhan Anda.

Untuk menghapus status, Anda dapat menggunakan backspace, klik kanan dan pilih Hapus status, atau pilih Hapus pada toolbar Desain.



Seiring alur kerja bertambah, alur kerja Anda mungkin tidak cukup di kanvas. Anda dapat:

1. Gunakan kontrol pada panel samping untuk mengubah ukuran atau menutup panel.
2. Gunakan kontrol toolbar Desain di bagian atas Kanvas untuk memperbesar grafik alur kerja masuk atau keluar.



Jalankan alur kerja Anda

Setelah membuat atau mengedit alur kerja dengan Workflow Studio, Anda dapat menjalankannya dan melihat pelaksanaannya di konsol Step Functions.

Untuk menjalankan alur kerja di Workflow Studio

1. Dalam mode Desain, Kode, atau Config, pilih Execute.

   Kotak dialog Mulai eksekusi terbuka di tab baru.
2. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

   > **Note**
   >
   > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.
3. Pilih Mulai Eksekusi.
4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

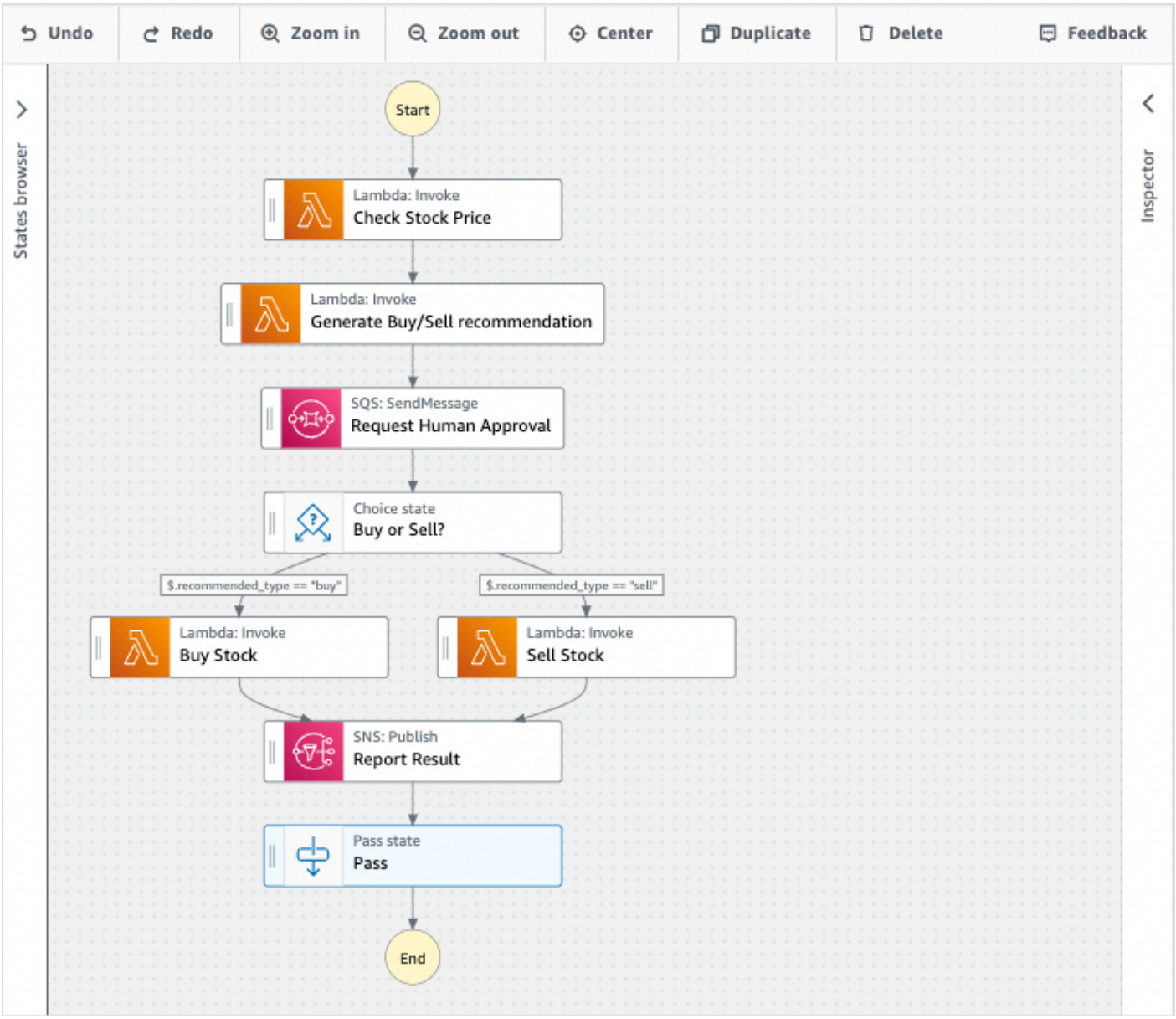
   Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Edit alur kerja

Anda dapat mengedit alur kerja yang ada secara visual di Mode desain Workflow Studio. Anda juga dapat mengedit definisi alur kerja di Mode kode Workflow Studio.

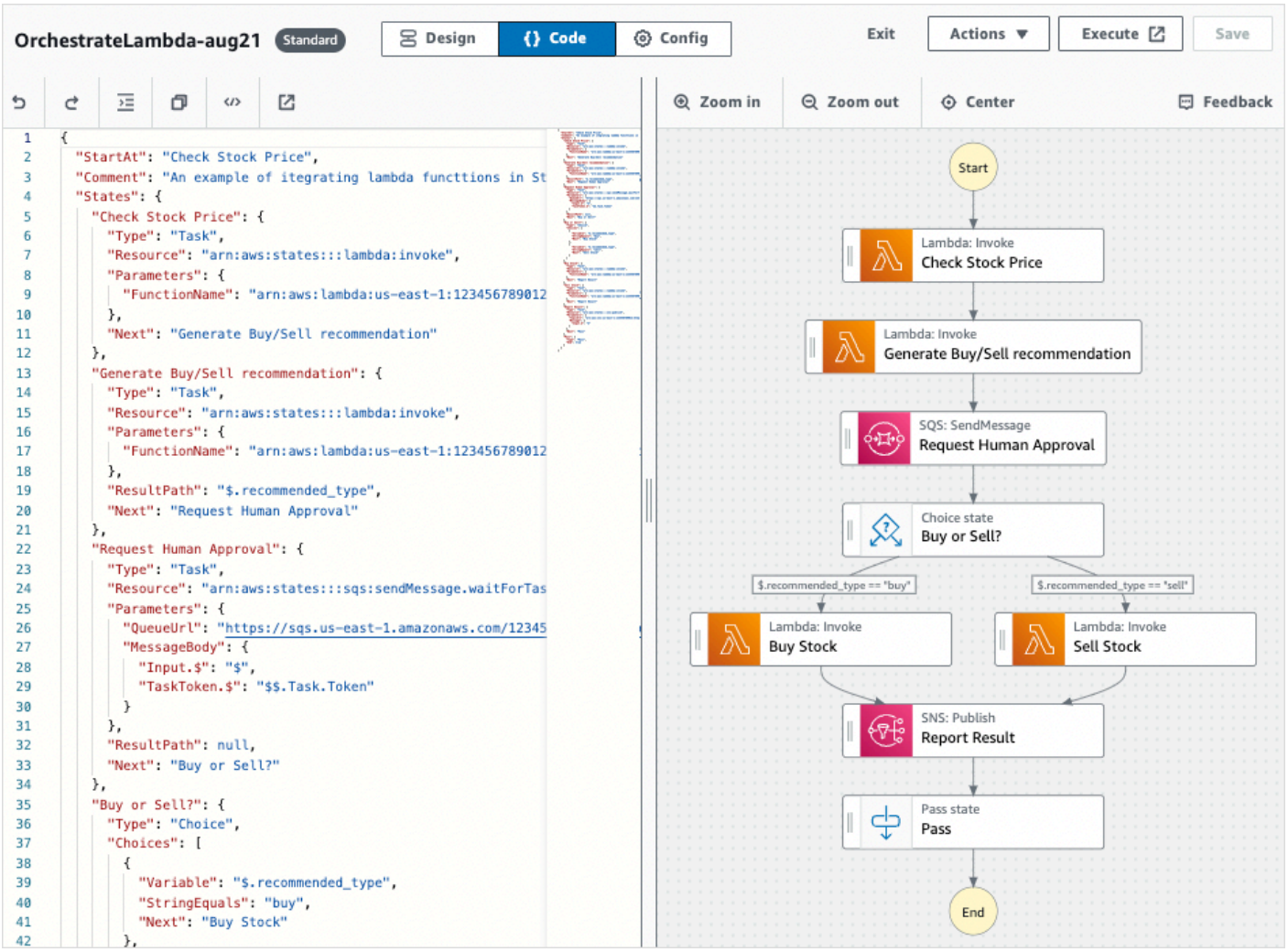
Untuk mengedit alur kerja yang ada:

1. Buka Konsol Step Functions.
2. Pada halaman Mesin negara, pilih alur kerja yang ingin Anda edit.
3. Pada halaman detail mesin Status, pilih Edit.
4. Alur kerja terbuka dalam mode Desain Workflow Studio. Edit alur kerja sesuai kebutuhan.



> **Note**
>
> Jika Anda melihat kesalahan dalam alur kerja Anda, Anda harus memperbaikinya dalam mode Desain. Anda tidak dapat beralih ke mode Kode atau Config jika ada kesalahan dalam alur kerja Anda.

5. (Opsional) Pilih Kode tombol untuk melihat atau mengedit definisi alur kerja di Workflow Studio.



6. Setelah selesai, pilih Simpan untuk menyimpan alur kerja yang diperbarui.
7. (Opsional) Untuk menjalankan alur kerja Anda yang diperbarui, pilih Jalankan. Kotak dialog Mulai eksekusi terbuka di tab baru.

Ekspor alur kerja Anda

Anda dapat mengekspor definisi alur kerja Amazon States Language (ASL) dan grafik alur kerja Anda:

1. Pilih alur kerja Anda di konsol Step Functions.
2. Pada halaman detail mesin Status, pilih Edit.
3. (Opsional) Alur kerja Anda terbuka dalam mode Desain Workflow Studio. Edit alur kerja Anda dalam mode Desain atau beralih ke mode Kode.
4. Pilih tombol dropdown Actions, lalu lakukan salah satu atau kedua hal berikut:

- Untuk mengekspor grafik alur kerja ke file SVG atau PNG, di bawah Ekspor grafik, pilih format yang Anda inginkan.
- Untuk mengekspor definisi alur kerja sebagai file JSON atau YAMG, di bawah definisi Ekspor, pilih format yang Anda inginkan.

Buat prototipe alur kerja Anda

Anda dapat menggunakan Workflow Studio untuk membuat prototipe alur kerja baru yang berisi sumber daya placeholder. Anda juga dapat membangun alur kerja Anda menggunakan Workflow Studio in. Application Composer Untuk membuat prototipe:

1. Masuk ke Konsol Step Functions.
2. Pilih Buat mesin status.
3. Dalam kotak dialog Pilih templat, pilih Kosong.
4. Pilih Pilih. Ini membuka Workflow Studio diMode desain.
5. Mode Desain Workflow Studio terbuka. Rancang alur kerja Anda di Workflow Studio. Untuk memasukkan sumber daya placeholder:
   a. Pilih status yang ingin Anda sertakan sumber daya placeholder, lalu di Konfigurasi:
      - Untuk status Lambda Invoke, pilih Nama fungsi, lalu pilih Masukkan nama fungsi. Anda juga dapat memasukkan nama khusus untuk fungsi Anda.
      - Untuk status Kirim Pesan Amazon SQS, pilih URL Antrian, lalu pilih Masukkan URL antrian. Masukkan URL antrean placeholder.
      - Untuk status Publikasikan Amazon SNS, dari Topik, pilih topik ARN.
      - Untuk semua status lain yang terdaftar dalam Tindakan, Anda dapat menggunakan konfigurasi default.

      > **Note**
      >
      > Jika Anda melihat kesalahan dalam alur kerja Anda, Anda harus memperbaikinya dalam mode Desain. Anda tidak dapat beralih ke mode Kode atau Config jika ada kesalahan dalam alur kerja Anda.

   b. (Opsional) Untuk melihat definisi ASL yang dibuat secara otomatis dari alur kerja Anda, pilih Definisi.
   c. (Opsional) Untuk memperbarui definisi alur kerja di Workflow Studio, pilih kode tombol.

> **Note**
>
> Jika Anda melihat kesalahan dalam definisi alur kerja Anda, Anda harus memperbaikinya dalam mode Kode. Anda tidak dapat beralih ke mode Desain atau Konfigurasi jika ada kesalahan dalam definisi alur kerja Anda.

6. (Opsional) Untuk mengedit nama mesin negara, pilih ikon edit di sebelah nama mesin status default MyStateMachinedan tentukan nama di kotak Nama mesin negara.

   Anda juga dapat beralih ke Modus Config untuk mengedit nama mesin status default.

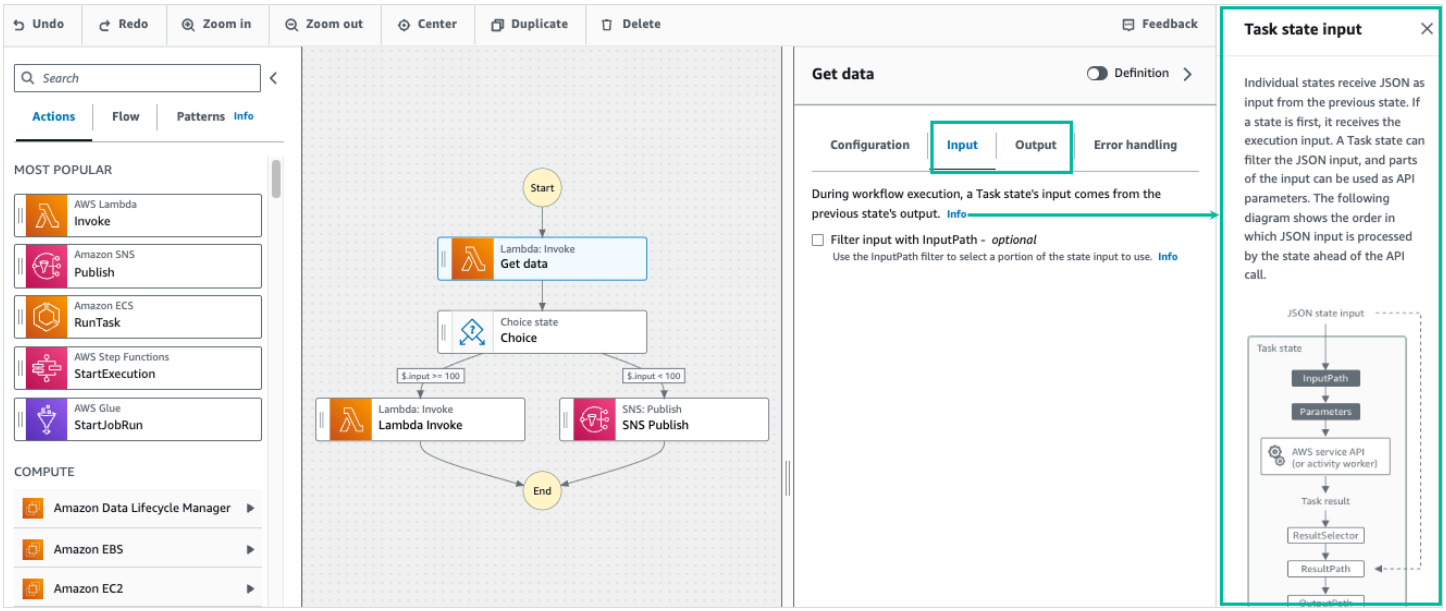
7. Tentukan pengaturan alur kerja Anda, seperti jenis mesin status dan peran pelaksanaannya.
8. Pilih Buat.

Anda sekarang telah membuat alur kerja baru dengan sumber daya placeholder yang dapat digunakan untuk prototipe. Anda dapat mengekspor definisi alur kerja dan grafik alur kerja Anda.

- Untuk mengekspor definisi alur kerja Anda sebagai file JSON atau YAMG, dalam mode Desain atau Kode, pilih tombol dropdown Tindakan. Kemudian, di bawah Definisi ekspor, pilih format yang ingin Anda ekspor. Anda dapat menggunakan definisi yang diekspor ini sebagai titik awal untuk pengembangan lokal dengan. AWS Toolkit for Visual Studio Code
- Untuk mengekspor grafik alur kerja Anda ke file SVG atau PNG, dalam mode Desain atau Kode, pilih tombol dropdown Tindakan. Kemudian, di bawah Definisi ekspor, pilih format yang Anda inginkan.

Konfigurasikan input dan output untuk status Anda

Setiap status membuat keputusan atau melakukan tindakan berdasarkan input yang diterimanya. Dalam kebanyakan kasus, kemudian output diteruskan ke status lain. Di Workflow Studio, Anda dapat mengonfigurasi bagaimana status memfilter dan memanipulasi data input dan outputnya di tab Input dan Output panel. Inspector Gunakan tautan Info untuk mengakses bantuan kontekstual saat mengonfigurasi input dan output.



Untuk informasi detail tentang bagaimana Step Functions memproses input dan output, lihat Pengolahan Input dan Output di Step Functions.

Konfigurasikan input ke status

Setiap status menerima masukan dari status sebelumnya sebagai JSON. Jika Anda ingin memfilter input, Anda dapat menggunakan `InputPath` filter di bawah tab Input di Inspector panel. `InputPath` adalah string, dimulai dengan $, yang mengidentifikasi simpul JSON tertentu. Ini disebut jalur referensi, dan mereka mengikuti JsonPath sintaks.

Untuk memfilter input:

- Pilih Filter input dengan InputPath.
- Masukkan yang valid JsonPathuntuk `InputPath` filter. Misalnya, **`$.data`**.

Filter `InputPath` akan ditambahkan ke alur kerja Anda.

Example Contoh 1: Gunakan InputPath filter di Workflow Studio

Katakanlah input ke status Anda mencakup data JSON berikut.

```
{
  "comment": "Example for InputPath",
  "dataset1": {
    "val1": 1,
    "val2": 2,
    "val3": 3
  },
  "dataset2": {
    "val1": "a",
    "val2": "b",
    "val3": "c"
  }
}
```

Untuk menerapkan `InputPath` filter, pilih Filter input dengan InputPath, lalu masukkan jalur referensi yang sesuai. Jika Anda masuk**`$.dataset2.val1`**, JSON berikut diteruskan sebagai masukan ke status.

```
{"a"}
```

Jalur referensi juga dapat memiliki pilihan nilai. Jika data yang Anda referensikan adalah `{ "a": [1, 2, 3, 4] }` dan Anda menerapkan jalur referensi `$.a[0:2]` sebagai `InputPath` filter, berikut ini adalah hasilnya.

```
[ 1, 2 ]
```

Paralel,Map, dan status Diteruskan aliran memiliki opsi pemfilteran input tambahan yang disebut `Parameters` di bawah tab Input mereka. Filter ini berlaku setelah InputPath filter dan dapat digunakan untuk membangun objek JSON kustom yang terdiri dari satu atau lebih pasangan kunci-nilai. Nilai masing-masing pasangan dapat berupa nilai statis, dapat dipilih dari input, atau dapat dipilih dari Objek konteks dengan jalur.

> **Note**
>
> Untuk menentukan bahwa parameter menggunakan jalur referensi untuk menunjuk ke simpul JSON di input, nama parameter harus diakhiri dengan `.$`.

Example Contoh 2: Buat input JSON kustom untuk status Paralel

Katakanlah data JSON berikut adalah input ke keadaan Paralel.

```
{
  "comment": "Example for Parameters",
  "product": {
    "details": {
      "color": "blue",
      "size": "small",
      "material": "cotton"
    },
    "availability": "in stock",
    "sku": "2317",
    "cost": "$23"
  }
}
```

Untuk memilih bagian dari input ini dan meneruskan pasangan nilai kunci tambahan dengan nilai statis, Anda dapat menentukan yang berikut di bidang Parameter, di bawah tab Input status Paralel.

```
{
 "comment": "Selecting what I care about.",
 "MyDetails": {
    "size.$": "$.product.details.size",
    "exists.$": "$.product.availability",
    "StaticValue": "foo"
    }
 }
```

Data JSON berikut akan menjadi hasilnya.

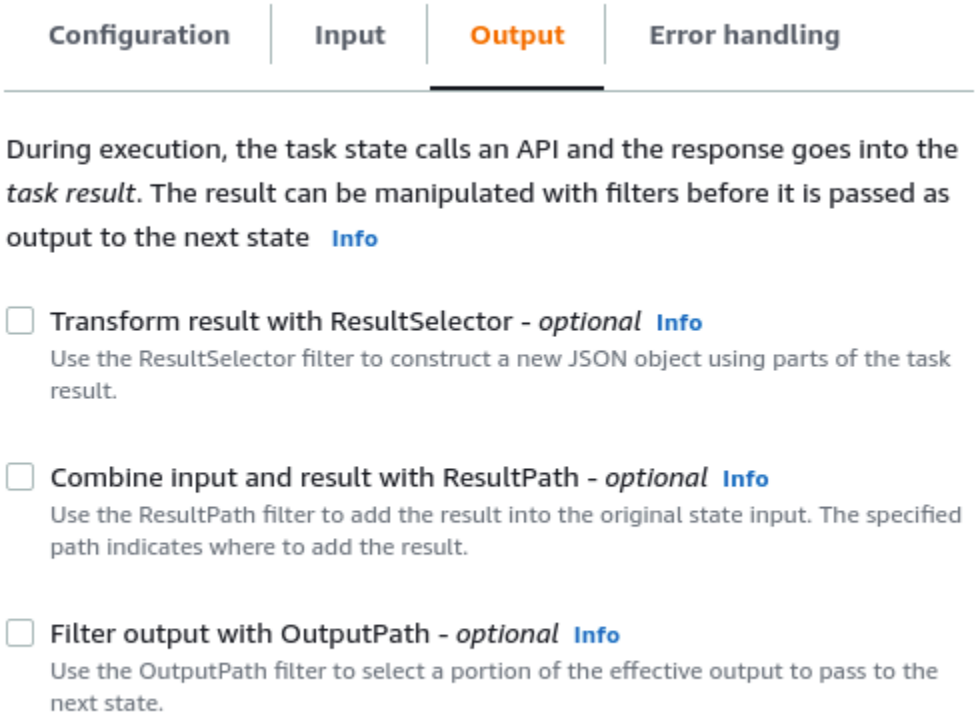
```
{
  "comment": "Selecting what I care about.",
```

```
  "MyDetails": {
    "size": "small",
    "exists": "in stock",
    "StaticValue": "foo"
  }
}
```

Konfigurasikan output status

Setiap status menghasilkan output JSON yang dapat difilter sebelum diteruskan ke status berikutnya. Ada beberapa filter yang tersedia, dan masing-masing mempengaruhi output dengan cara yang berbeda. Filter keluaran yang tersedia untuk setiap status tercantum di bawah tab Output di panel Inspector. Untuk Tugas negara bagian, filter keluaran apa pun yang Anda pilih diproses dalam urutan ini:

1. `ResultSelector`: Gunakan filter ini untuk memanipulasi hasil status. Anda dapat membangun sebuah objek JSON baru dengan bagian dari hasil.
2. `ResultPath`: Gunakan filter ini untuk memilih kombinasi input status dan hasil tugas untuk diteruskan ke output.
3. `OutputPath`: Gunakan filter ini untuk memfilter output JSON untuk memilih informasi mana dari hasil yang akan diteruskan ke status berikutnya.



Gunakan ResultSelector

`ResultSelector` adalah filter output opsional untuk status berikut:

- Tugasnegara bagian, yang merupakan semua status yang tercantum di tab Tindakan padaPeramban status.
- Mapstates, di tab Flow pada browser States.
- Paralelstates, di tab Flow pada browser States.

`ResultSelector` dapat digunakan untuk membangun sebuah objek JSON kustom yang terdiri atas satu pasangan nilai kunci atau lebih. Nilai masing-masing pasangan dapat berupa nilai statis atau dipilih dari hasil status dengan jalur.

> **Note**
>
> Untuk menentukan bahwa parameter menggunakan jalur untuk mereferensikan simpul JSON dalam hasil, nama parameter harus diakhiri dengan `.$`.

Example Contoh untuk menggunakan ResultSelector filter

Dalam contoh ini, Anda gunakan `ResultSelector` untuk memanipulasi respons dari panggilan Amazon CreateCluster EMR API untuk status `CreateCluster` EMR Amazon. Berikut ini adalah hasil dari panggilan Amazon EMR `CreateCluster` API.

```
{
  "resourceType": "elasticmapreduce",
  "resource": "createCluster.sync",
  "output": {
    "SdkHttpMetadata": {
      "HttpHeaders": {
        "Content-Length": "1112",
        "Content-Type": "application/x-amz-JSON-1.1",
        "Date": "Mon, 25 Nov 2019 19:41:29 GMT",
        "x-amzn-RequestId": "1234-5678-9012"
      },
      "HttpStatusCode": 200
    },
    "SdkResponseMetadata": {
      "RequestId": "1234-5678-9012"
    },
    "ClusterId": "AKIAIOSFODNN7EXAMPLE"
  }
}
```

Untuk memilih bagian dari informasi ini dan meneruskan pasangan kunci-nilai tambahan dengan nilai statis, tentukan yang berikut ini di ResultSelectorbidang, di bawah tab Output negara bagian.

```
{
 "result": "found",
 "ClusterId.$": "$.output.ClusterId",
 "ResourceType.$": "$.resourceType"
 }
```

Menggunakan `ResultSelector` menghasilkan hasil sebagai berikut.

```
{
 "result": "found",
 "ClusterId": "AKIAIOSFODNN7EXAMPLE",
 "ResourceType": "elasticmapreduce"
```

```
}
```

Gunakan ResultPath

Output dari status dapat menjadi salinan dari input, hasil yang dibuat , atau kombinasi dari input dan hasilnya. Gunakan `ResultPath` untuk mengontrol kombinasi ini yang akan diteruskan ke output status. Untuk kasus penggunaan `ResultPath` lebih banyak, lihat ResultPath.

`ResultPath` adalah filter output opsional untuk status berikut:

- Tugasnegara bagian, yang merupakan semua status yang tercantum di tab Tindakan pada browser Negara.
- Mapstates, di tab Flow pada browser States.
- Paralelstates, di tab Flow pada browser States.
- Diteruskanstates, di tab Flow pada browser States.

`ResultPath` dapat digunakan untuk menambahkan hasilnya ke input status asli. Jalur yang ditentukan menunjukkan tempat untuk menambahkan hasilnya.

Example Contoh untuk menggunakan ResultPath filter

Katakanlah berikut ini adalah masukan ke status Tugas.

```
{
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

Hasil dari status Tugas adalah sebagai berikut.

```
Hello, AWS Step Functions
```

Anda dapat menambahkan hasil ini ke input status dengan menerapkan `ResultPath` dan memasukkan jalur referensi yang menunjukkan tempat untuk menambahkan hasilnya, seperti `$.taskresult`:

Dengan `ResultPath` ini, berikut adalah JSON yang diteruskan sebagai output status.

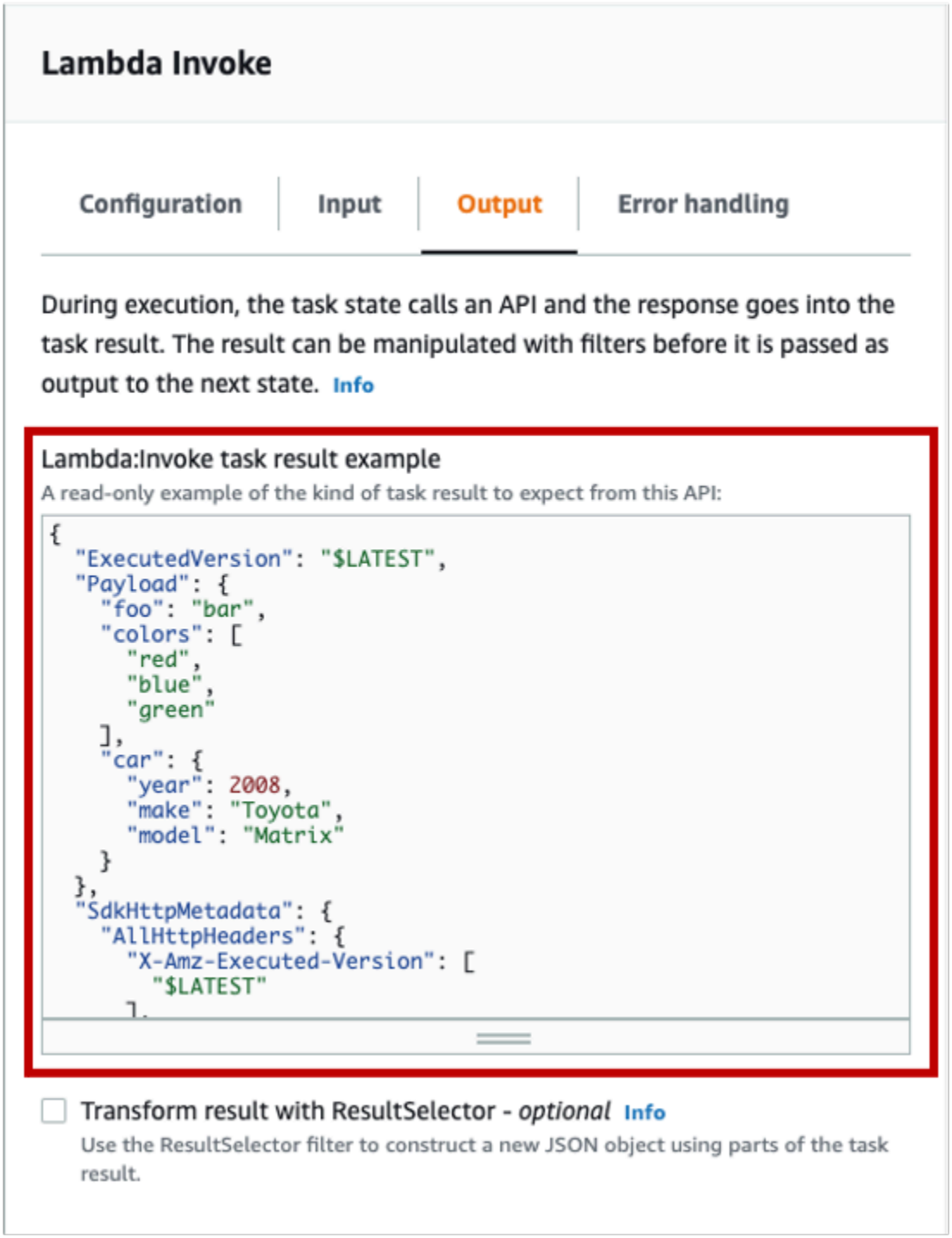
```
{
  "details": "Default example",
  "who": "AWS Step Functions",
  "taskresult": "Hello, AWS Step Functions!"
}
```

Gunakan OutputPath

Filter `OutputPath` memungkinkan Anda memfilter informasi yang tidak diinginkan, dan hanya meneruskan bagian JSON yang Anda pedulikan. `OutputPath`Ini adalah string, dimulai dengan`$`, yang mengidentifikasi node dalam teks JSON.

Example Contoh untuk menggunakan OutputPath filter

Panggilan API Lambda Invoke mengembalikan metadata selain muatan, yang merupakan hasil fungsi Lambda ini. Contoh respons dari panggilan API ini ditampilkan di bawah tab Output status.



Anda dapat menggunakan `OutputPath` untuk memfilter metadata tambahan. Secara default, nilai OutputPathfilter untuk status Lambda Invoke yang dibuat melalui Workflow Studio adalah. `$.Payload` Nilai default ini menghapus metadata tambahan dan mengembalikan output yang setara dengan menjalankan fungsi Lambda secara langsung.

Contoh hasil tugas Lambda Invoke dan nilai `$.Payload` untuk filter Output meneruskan data JSON berikut sebagai output.

```
{
 "foo": "bar",
 "colors": [
      "red",
      "blue",
      "green"
 ],
 "car": {
      "year": 2008,
      "make": "Toyota",
      "model": "Matrix"
 }
}
```

> **Note**
>
> Karena `OutputPath` filter adalah filter keluaran terakhir yang berlaku, jika Anda menggunakan filter keluaran tambahan seperti `ResultSelector` atau`ResultPath`, Anda harus memodifikasi nilai default `$.Payload` untuk `OutputPath` filter yang sesuai.

Peran eksekusi di Workflow Studio

Setiap mesin Step Functions status memerlukan peran AWS Identity and Access Management (IAM) yang memberikan izin mesin status untuk melakukan tindakan Layanan AWS dan sumber daya atau memanggil API pihak ketiga. Peran ini disebut peran eksekusi. Peran ini harus berisi IAM kebijakan untuk setiap tindakan, misalnya kebijakan yang memungkinkan mesin status menjalankan AWS Lambda fungsi, menjalankan AWS Batch pekerjaan, atau memanggil Stripe API. Step Functionsmengharuskan Anda untuk memberikan peran eksekusi dalam kasus-kasus berikut:

- Anda membuat mesin status di konsol, AWS SDK, atau AWS CLI menggunakan CreateStateMachineAPI.
- Anda menguji status di konsol, AWS SDK, atau AWS CLI menggunakan TestStateAPI.

Workflow Studio memiliki kemampuan yang memudahkan pengelolaan peran eksekusi untuk alur kerja Anda.

Topik

- Tentang peran yang dibuat secara otomatis
- Secara otomatis menghasilkan peran
- Menyelesaikan masalah pembuatan peran
- Peran untuk menguji Tugas HTTP di Workflow Studio
- Peran untuk menguji integrasi layanan yang dioptimalkan di Workflow Studio
- Peran untuk menguji integrasi layanan AWS SDK di Workflow Studio
- Peran untuk menguji status alur di Workflow Studio

Tentang peran yang dibuat secara otomatis

Saat Anda membuat mesin status di Step Functions konsol, Workflow Studio dapat secara otomatis membuat peran eksekusi untuk Anda yang berisi IAM kebijakan yang diperlukan. Workflow Studio menganalisis definisi mesin status Anda dan menghasilkan kebijakan dengan hak istimewa paling sedikit yang diperlukan untuk menjalankan alur kerja Anda.

Workflow Studio dapat menghasilkan IAM kebijakan untuk hal-hal berikut:

- Tugas HTTP yang memanggil API pihak ketiga.
- Status tugas yang memanggil lainnya Layanan AWS menggunakan integrasi yang dioptimalkan, seperti LambdaInvoke, DynamoDB GetItem, atau. AWS Glue StartJobRun
- Status tugas yang menjalankan alur kerja bersarang.
- Status Peta Terdistribusi, termasuk kebijakan untuk memulai eksekusi alur kerja anak, mencantumkan Amazon S3 bucket, dan membaca atau menulis objek S3.
- X-Raymenelusuri. Setiap peran yang dibuat secara otomatis di Workflow Studio berisi kebijakan yang memberikan izin untuk mesin status untuk mengirim jejak. X-Ray
- Logging menggunakanCloudWatchLogsaat logging diaktifkan pada mesin status.

Workflow Studio tidak dapat membuat IAM kebijakan untuk status Tugas yang memanggil lainnya Layanan AWS menggunakan integrasi AWS SDK.

Secara otomatis menghasilkan peran

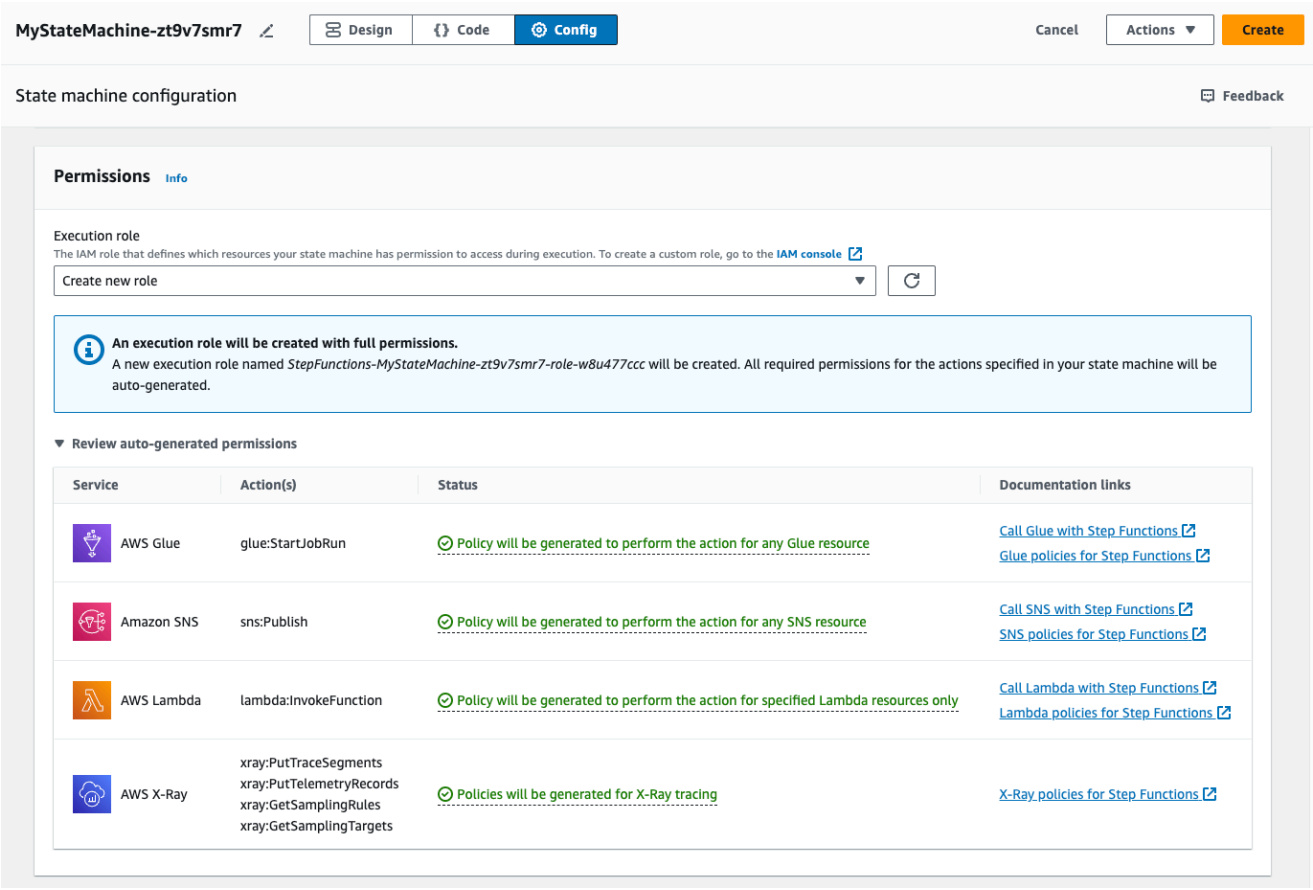
1. Buka Konsol Step Functions dan pilih Buat mesin status.

Anda juga dapat memperbarui mesin status yang ada. Lihat Langkah 4 jika Anda memperbarui mesin status.

2. Dalam kotak dialog Pilih templat, pilih Kosong.
3. Pilih Pilih. Ini membuka Workflow Studio diMode desain.
4. Pilih tab Config.
5. Gulir ke bawah ke bagian Izin, dan lakukan hal berikut:

   a. Untuk peran Eksekusi, pastikan Anda menyimpan pilihan default Buat peran baru.

   Workflow Studio secara otomatis menghasilkan semua IAM kebijakan yang diperlukan untuk setiap status valid dalam definisi mesin status Anda. Ini menampilkan spanduk dengan pesan, Peran eksekusi akan dibuat dengan izin penuh.



> **Tip**
>
> Untuk meninjau izin yang dibuat secara otomatis Workflow Studio untuk mesin status Anda, pilih Tinjau izin yang dibuat secara otomatis.

> **Note**
>
> Jika Anda menghapus IAM role yang Step Functions buat, Step Functions tidak dapat membuatnya kembali nanti. Demikian pula, jika Anda mengubah peran (misalnya, dengan menghapus Step Functions dari principal dalam kebijakan IAM), Step Functions tidak dapat memulihkan pengaturan aslinya nanti.

Jika Workflow Studio tidak dapat membuat semua IAM kebijakan yang diperlukan, itu akan menampilkan spanduk dengan pesan Izin untuk tindakan tertentu tidak dapat dibuat secara otomatis. IAMPeran akan dibuat dengan izin sebagian saja. Untuk informasi tentang cara menambahkan izin yang hilang, lihatMenyelesaikan masalah pembuatan peran.

b. Pilih Buat jika Anda membuat mesin status. Jika tidak, pilih Simpan.

c. Pilih Konfirmasi di kotak dialog yang muncul.

Workflow Studio menyimpan mesin status Anda dan membuat peran eksekusi baru.

Menyelesaikan masalah pembuatan peran

Workflow Studio tidak dapat secara otomatis menghasilkan peran eksekusi dengan semua izin yang diperlukan dalam kasus berikut:

- Ada kesalahan di mesin negara Anda. Pastikan untuk menyelesaikan semua kesalahan validasi di Workflow Studio. Juga, pastikan bahwa Anda mengatasi kesalahan sisi server yang Anda temui selama penyimpanan.
- Mesin status Anda berisi tugas menggunakan integrasi AWS SDK. Workflow Studio tidak dapat membuat IAM kebijakan secara otomatis dalam kasus ini. Workflow Studio menampilkan spanduk dengan pesan, Izin untuk tindakan tertentu tidak dapat dibuat secara otomatis. IAMPeran akan dibuat dengan izin sebagian saja. Di tabel Meninjau izin yang dibuat secara otomatis, pilih konten di Status untuk informasi selengkapnya tentang kebijakan yang tidak ada peran eksekusi Anda.

Workflow Studio masih dapat menghasilkan peran eksekusi, tetapi peran ini tidak akan berisi IAM kebijakan untuk semua tindakan. Lihat tautan di bawah tautan Dokumentasi untuk menulis kebijakan Anda sendiri dan menambahkannya ke peran setelah dibuat. Tautan ini tersedia bahkan setelah Anda menyimpan mesin negara.

Peran untuk menguji Tugas HTTP di Workflow Studio

Anda memerlukan peran eksekusi untuk menguji status Tugas HTTP. Jika Anda tidak memiliki peran dengan izin yang memadai, gunakan salah satu opsi berikut untuk membuat peran:

- Buat peran secara otomatis dengan Workflow Studio (disarankan) - Ini adalah opsi aman. Tutup kotak dialog Test state dan ikuti instruksi diSecara otomatis menghasilkan peran. Ini akan mengharuskan Anda untuk membuat atau memperbarui mesin status Anda terlebih dahulu, kemudian kembali ke Workflow Studio untuk menguji status Anda.
- Gunakan peran dengan akses Administrator — Jika Anda memiliki izin untuk membuat peran dengan akses penuh ke semua layanan dan sumber daya AWS, Anda dapat menggunakan peran tersebut untuk menguji semua jenis status dalam alur kerja Anda. Untuk melakukannya, Anda dapat membuat peran Step Functions layanan dan menambahkan AdministratorAccess kebijakan ke dalamnya di IAM konsol https://console.aws.amazon.com/iam/.

Peran untuk menguji integrasi layanan yang dioptimalkan di Workflow Studio

Anda memerlukan peran eksekusi untuk status Tugas yang memanggil integrasi layanan yang dioptimalkan. Jika Anda tidak memiliki peran dengan izin yang memadai, gunakan salah satu opsi berikut untuk membuat peran:

- Gunakan tautan dokumentasi di Workflow Studio untuk menulis IAM kebijakan Anda sendiri (disarankan) — Ini adalah opsi aman. Tutup kotak dialog Test state dan ikuti instruksi diSecara otomatis menghasilkan peran. Ini akan mengharuskan Anda untuk membuat atau memperbarui mesin status Anda terlebih dahulu, kemudian kembali ke Workflow Studio untuk menguji status Anda.
- Gunakan peran dengan akses Administrator — Jika Anda memiliki izin untuk membuat peran dengan akses penuh ke semua layanan dan sumber daya AWS, Anda dapat menggunakan peran tersebut untuk menguji semua jenis status dalam alur kerja Anda. Untuk melakukannya, Anda

dapat membuat peran Step Functions layanan dan menambahkan AdministratorAccess kebijakan ke dalamnya di IAM konsol https://console.aws.amazon.com/iam/.

Peran untuk menguji integrasi layanan AWS SDK di Workflow Studio

Anda memerlukan peran eksekusi ke status Tugas yang memanggil integrasi AWS SDK. Jika Anda tidak memiliki peran dengan izin yang memadai, gunakan salah satu opsi berikut untuk membuat peran:

- Gunakan tautan dokumentasi di Workflow Studio untuk menulis IAM kebijakan Anda sendiri (disarankan) — Ini adalah opsi aman. Tutup kotak dialog Test state dan ikuti instruksi diSecara otomatis menghasilkan peran. Ini akan mengharuskan Anda untuk membuat atau memperbarui mesin status Anda terlebih dahulu, kemudian kembali ke Workflow Studio untuk menguji status Anda. Lakukan hal-hal berikut:
  1. Tutup kotak dialog Test state
  2. Pilih tab Config untuk melihat mode Config.
  3. Gulir ke bawah ke bagian Izin.
  4. Workflow Studio menampilkan spanduk dengan pesan, Izin untuk tindakan tertentu tidak dapat dibuat secara otomatis. IAMPeran akan dibuat dengan izin sebagian saja. Pilih Tinjau izin yang dibuat secara otomatis.
  5. Tabel izin yang dibuat secara otomatis akan menampilkan baris yang menunjukkan tindakan yang sesuai dengan status tugas yang ingin Anda uji. Lihat tautan di bawah tautan Dokumentasi untuk menulis IAM kebijakan Anda sendiri ke dalam peran khusus.
- Gunakan peran dengan akses Administrator — Jika Anda memiliki izin untuk membuat peran dengan akses penuh ke semua layanan dan sumber daya AWS, Anda dapat menggunakan peran tersebut untuk menguji semua jenis status dalam alur kerja Anda. Untuk melakukannya, Anda dapat membuat peran Step Functions layanan dan menambahkan AdministratorAccess kebijakan ke dalamnya di IAM konsol https://console.aws.amazon.com/iam/.

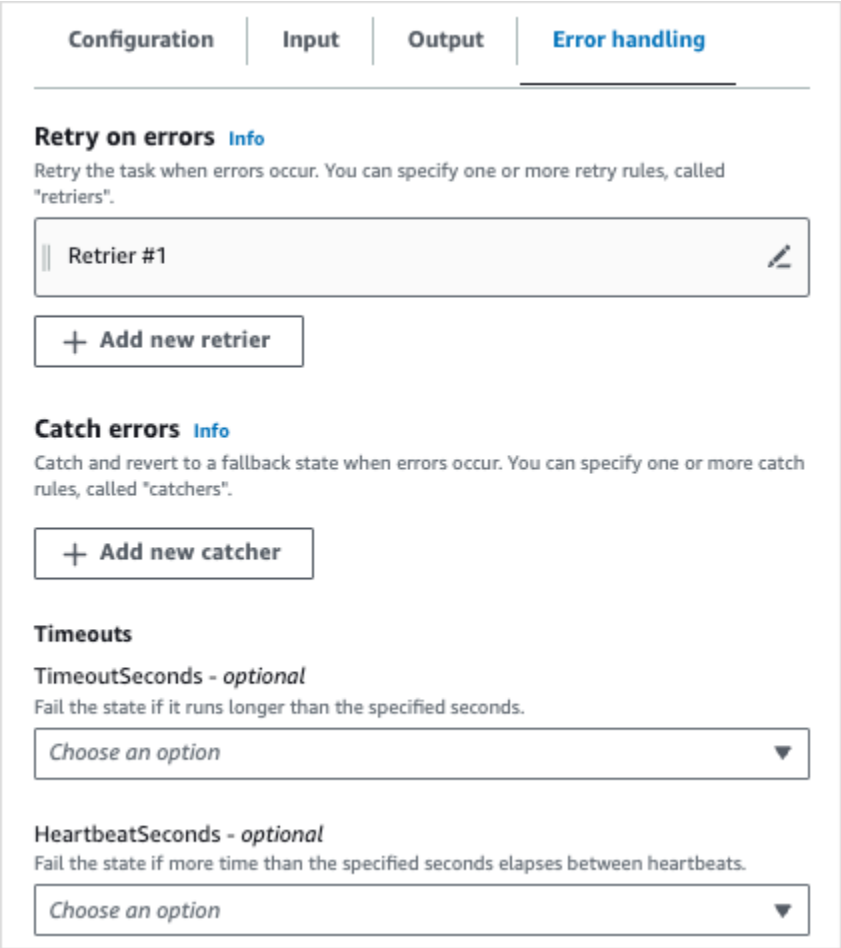
Peran untuk menguji status alur di Workflow Studio

Anda memerlukan peran eksekusi untuk menguji status alur di Workflow Studio. Status aliran adalah keadaan yang mengarahkan aliran eksekusi, sepertiPilihan,, ParalelMap,Diteruskan,Tunggu,Berhasil, atauGagal. TestStateAPI tidak berfungsi dengan status Peta atau Paralel. Gunakan salah satu opsi berikut untuk membuat peran untuk menguji status aliran:

- Gunakan peran apa pun dalam Akun AWS (disarankan) Anda — Status aliran tidak memerlukan IAM kebijakan khusus apa pun, karena tidak memanggil AWS tindakan atau sumber daya. Karena itu, Anda dapat menggunakan IAM peran apa pun dalam peran Anda Akun AWS.
  1. Dalam kotak dialog Status uji, pilih peran apa pun dari daftar tarik-turun peran Eksekusi.
  2. Jika tidak ada peran yang muncul di daftar dropdown, lakukan hal berikut:
     a. Di IAM konsol https://console.aws.amazon.com/iam/, pilih Peran.
     b. Pilih peran dari daftar, dan salin ARN dari halaman detail peran. Anda harus memberikan ARN ini di kotak dialog status Uji.
     c. Dalam kotak dialog Status uji, pilih Masukkan ARN peran dari daftar tarik-turun peran Eksekusi.
     d. Tempelkan ARN di Peran ARN.
- Gunakan peran dengan akses Administrator — Jika Anda memiliki izin untuk membuat peran dengan akses penuh ke semua layanan dan sumber daya AWS, Anda dapat menggunakan peran tersebut untuk menguji semua jenis status dalam alur kerja Anda. Untuk melakukannya, Anda dapat membuat peran Step Functions layanan dan menambahkan AdministratorAccess kebijakan ke dalamnya di IAM konsol https://console.aws.amazon.com/iam/.

Penanganan kesalahan

Secara default, ketika status melaporkan kesalahan, Step Functions menyebabkan eksekusi alur kerja gagal sepenuhnya. Untuk tindakan dan beberapa status alur, Anda dapat mengonfigurasi cara Step Functions menangani kesalahan. Bahkan jika Anda telah mengonfigurasi penanganan kesalahan, beberapa kesalahan mungkin masih menyebabkan eksekusi alur kerja gagal. Untuk informasi selengkapnya, lihat Penanganan kesalahan dalam Step Functions. Di Workflow Studio, konfigurasikan penanganan kesalahan di tab Penanganan kesalahan Inspector panel.



Coba lagi pada kesalahan

Anda dapat menambahkan satu aturan atau lebih ke status tindakan dan status alur Paralel untuk mencoba lagi tugas ketika terjadi kesalahan. Aturan-aturan ini disebut retriers. Untuk menambahkan retrier, pilih ikon edit di kotak Retrier #1, lalu konfigurasikan opsinya:

- (Opsional) Di kolom Komentar, tambahkan komentar Anda. Ini tidak akan memengaruhi alur kerja, tetapi dapat digunakan untuk membuat anotasi alur kerja Anda.
- Tempatkan kursor di bidang Kesalahan dan pilih kesalahan yang akan memicu retrier, atau masukkan nama kesalahan khusus. Anda dapat memilih atau menambahkan beberapa kesalahan.
- (Opsional) Atur Interval. Ini adalah waktu dalam detik sebelum Step Functions melakukan percobaan ulang pertama. Percobaan ulang tambahan akan mengikuti interval yang dapat Anda konfigurasikan dengan Percobaan maks dan Tingkat backoff.
- (Opsional) Atur Percobaan maks. Ini adalah jumlah maksimum percobaan ulang sebelum Step Functions akan menyebabkan eksekusi gagal.

- (Opsional) Atur Tingkat backoff. Ini adalah pengganda yang menentukan berapa banyak interval percobaan ulang akan meningkat dengan setiap usaha.

> **Note**
>
> Tidak semua opsi penanganan kesalahan tersedia untuk semua status. Lambda Invoke memiliki satu retrier yang dikonfigurasi secara default.

Tangkap kesalahan

Anda dapat menambahkan satu atau beberapa aturan ke status tindakan Paralel dan ke status and Map flow untuk menangkap kesalahan. Aturan-aturan ini disebut catcher. Untuk menambahkan catcher, pilih Tambahkan catcher baru, lalu konfigurasikan opsinya:

- (Opsional) Di kolom Komentar, tambahkan komentar Anda. Ini tidak akan memengaruhi alur kerja, tetapi dapat digunakan untuk membuat anotasi alur kerja Anda.
- Tempatkan kursor di bidang Kesalahan dan pilih kesalahan yang akan memicu penangkap, atau masukkan nama kesalahan khusus. Anda dapat memilih atau menambahkan beberapa kesalahan.
- Di bidang status Fallback, pilih status mundur. Ini adalah status saat alur kerja akan berpindah ke berikutnya, setelah kesalahan tertangkap.
- (Opsional) Di ResultPathlapangan, tambahkan `ResultPath` filter untuk menambahkan kesalahan ke input status asli. `ResultPath`Harus valid JsonPath. Ini akan dikirim ke status fallback.

Timeout

Anda dapat mengonfigurasi waktu habis untuk status tindakan untuk menetapkan jumlah maksimum detik status Anda dapat menjalankan sebelum gagal. Gunakan waktu habis untuk mencegah eksekusi macet. Untuk mengonfigurasi waktu habis, masukkan jumlah detik status Anda harus menunggu sebelum eksekusi gagal. Untuk informasi selengkapnya tentang batas waktu, lihat `TimeoutSeconds` di Tugas negara bagian.

HeartbeatSeconds

Anda dapat mengonfigurasi Detak Jantung atau pemberitahuan berkala yang dikirim oleh tugas Anda. Jika Anda menetapkan interval heartbeat, dan status Anda tidak mengirim notifikasi heartbeat dalam interval yang dikonfigurasi, tugas ditandai sebagai gagal. Untuk mengonfigurasi heartbeat,

tetapkan jumlah detik dalam bilangan bulat positif, bukan nol. Untuk informasi lebih lanjut, lihat `HeartBeatSeconds` di Tugas negara bagian.

Tutorial: Belajar untuk menggunakan Studio Alur Kerja AWS Step Functions

Dalam tutorial ini, Anda akan mempelajari dasar-dasar bekerja dengan Workflow Studio untuk AWS Step Functions. Di Mode desain Workflow Studio, Anda akan membuat mesin status yang berisi beberapa status, termasuk`Pass`,,`Choice`, `FailWait`, dan`Parallel`. Anda akan menggunakan fitur seret dan lepas untuk mencari, memilih, dan mengonfigurasi status ini. Kemudian, Anda akan melihat definisi auto-generated Amazon States Language (ASL) dari alur kerja Anda. Anda juga akan menggunakan Mode kode Workflow Studio untuk mengedit definisi alur kerja. Kemudian, Anda akan keluar dari Workflow Studio, menjalankan state machine, dan meninjau detail eksekusi.

Dalam tutorial ini, Anda juga akan belajar cara memperbarui mesin status dan melihat perubahan dalam output eksekusi. Terakhir, Anda akan melakukan langkah pembersihan dan menghapus mesin status Anda.

Setelah Anda menyelesaikan tutorial ini, Anda akan tahu cara menggunakan Workflow Studio untuk membuat dan mengkonfigurasi alur kerja menggunakan mode Desain dan Kode. Anda juga akan tahu cara memperbarui, menjalankan, dan menghapus mesin status Anda.

> **Note**
>
> Sebelum Anda mulai, pastikan untuk menyelesaikan prasyarat untuk tutorial ini.

Topik

- Langkah 1: Arahkan ke Workflow Studio
- Langkah 2: Buat mesin negara
- Langkah 3: Tinjau definisi Bahasa Amazon States yang dibuat secara otomatis
- Langkah 4: Edit definisi alur kerja dalam mode Kode
- Langkah 5: Simpan mesin negara
- Langkah 6: Jalankan mesin negara
- Langkah 7: Perbarui mesin negara Anda
- Langkah 8: Membersihkan

Langkah 1: Arahkan ke Workflow Studio

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Dalam kotak dialog Pilih templat, pilih Kosong.
3. Pilih Pilih. Ini membuka Workflow Studio diMode desain.

Langkah 2: Buat mesin negara

Dalam Studio Alur Kerja, mesin status adalah representasi grafis dari alur kerja Anda. Dengan Workflow Studio, Anda dapat menentukan, mengonfigurasi, dan memeriksa setiap langkah alur kerja Anda. Pada langkah-langkah berikut, Anda menggunakan Mode desain Workflow Studio untuk membuat mesin status Anda.

Untuk membuat mesin status

1. Pastikan Anda berada dalam mode Desain Workflow Studio.
2. Dari Peramban status sebelah kiri, pilih tab Flow. Kemudian, seret status Pass ke status kosong berlabel Seret status pertama di sini.
3. Seret status Choice dari tab Flow dan jatuhkan di bawah status Pass.
4. Untuk nama Negara, ganti nama default Choice. Untuk tutorial ini, gunakan namanya**`IsHelloWorldExample`**.
5. Seret status Pass lain dan jatuhkan ke satu cabang IsHelloWorldExamplenegara bagian. Kemudian, seret status Fail dan jatuhkan di bawah cabang IsHelloWorldExamplenegara bagian lainnya.
6. Pilih status Pass (1), dan ganti namanya menjadi**`Yes`**. Ganti nama status Gagal sebagai**`No`**.
7. Tentukan logika percabangan IsHelloWorldExamplenegara menggunakan variabel boolean. `IsHelloWorldExample`

   Jika `IsHelloWorldExample` ya`False`, alur kerja akan masuk ke No state. Jika tidak, alur kerja akan melanjutkan alur eksekusinya dalam status Ya.

   Untuk menentukan logika percabangan, lakukan hal berikut:

   a. Pilih IsHelloWorldExamplestatus padaKanvas, dan kemudian di bawah Aturan Pilihan, pilih ikon edit di kotak Aturan #1 untuk menentukan aturan pilihan pertama.

   b. Pilih Tambahkan kondisi.

c. Dalam kotak dialog Ketentuan untuk aturan #1, masukkan **`$.IsHelloWorldExample`** di bawah Variabel.

d. Pilih sama dengan di bawah Operator.

e. Pilih konstanta Boolean di bawah Nilai, dan kemudian pilih true dari daftar dropdown.

f. Pilih Simpan kondisi.

g. Pastikan Then next state is: dropdown list has yes selected.

h. Pilih Tambahkan aturan pilihan baru, lalu pilih Tambahkan kondisi.

i. Dalam kotak Aturan #2, tentukan aturan pilihan kedua ketika nilai `IsHelloWorldExample` variabel salah dengan mengulangi sublangkah 7.c hingga 7.f. Untuk langkah 7.e, pilih false, bukan true.

j. Di kotak Rule #2, pilih No dari Then next state is: dropdown list.

k. Di kotak Aturan default, pilih ikon edit untuk menentukan aturan pilihan default, lalu pilih Ya dari daftar dropdown.

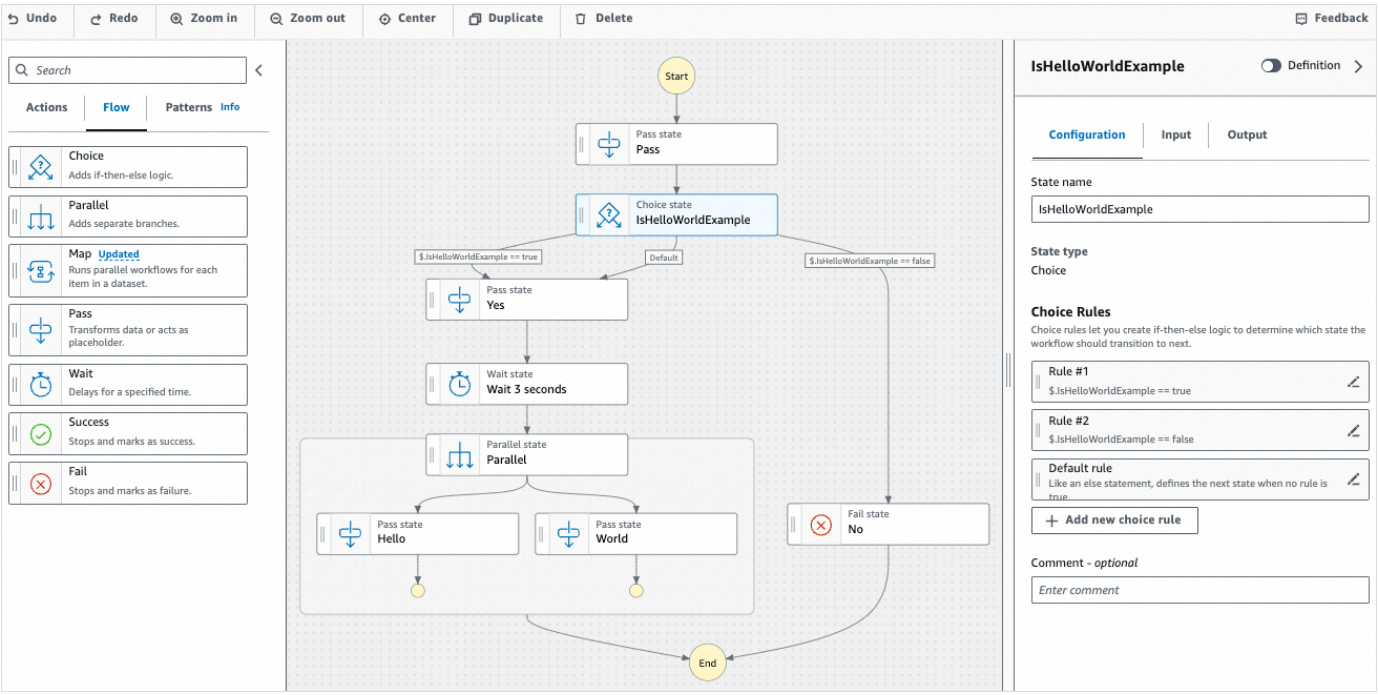
8. Tambahkan status Tunggu setelah status Ya, dan beri nama**`Wait 3 sec`**. Kemudian, konfigurasikan waktu tunggu menjadi tiga detik dengan melakukan langkah-langkah berikut:

   a. Di bawah Opsi, pertahankan pilihan default Tunggu interval tetap.

   b. Di bawah Detik, pastikan Enter detik dipilih, lalu masukkan **`3`** di dalam kotak.

9. Setelah status Tunggu 3 detik, tambahkan status Paralel. Tambahkan dua status Pass di dua cabangnya. Sebutkan status Pass pertama**`Hello`**. Sebutkan status Pass kedua**`World`**.

   Alur Kerja yang telah selesai akan terlihat seperti ini:



Langkah 3: Tinjau definisi Bahasa Amazon States yang dibuat secara otomatis

Saat Anda menyeret dan melepas status dari tab Flow ke kanvas, Workflow Studio secara otomatis menyusun definisi Amazon States Language (ASL) dari alur kerja Anda secara real-time. Di Inspector panel, pilih tombol sakelar Definisi untuk melihat definisi ini atau beralih ke Mode kode untuk mengedit definisi ini sesuai kebutuhan. Untuk informasi tentang mengedit definisi alur kerja, lihat Langkah 4 dari tutorial ini.

- (Opsional) Pilih Definisi pada panel Inspector dan lihat alur kerja state machine.

  Kode contoh berikut menunjukkan definisi Bahasa Amazon States yang dibuat secara otomatis untuk mesin `IsHelloWorldExample` status. `ChoiceStatus` yang Anda tambahkan di Workflow Studio digunakan untuk menentukan alur eksekusi berdasarkan logika percabangan yang Anda tentukan di Langkah 2.

```
{
  "Comment": "A Hello World example of the Amazon States Language using Pass
 states",
  "StartAt": "Pass",
  "States": {
    "Pass": {
```

```
    "Type": "Pass",
    "Next": "IsHelloWorldExample",
    "Comment": "A Pass state passes its input to its output, without performing
work. Pass states are useful when constructing and debugging state machines."
  },
  "IsHelloWorldExample": {
    "Type": "Choice",
    "Comment": "A Choice state adds branching logic to a state machine. Choice
rules can implement 16 different comparison operators, and can be combined using
And, Or, and Not\"",
    "Choices": [
      {
        "Variable": "$.IsHelloWorldExample",
        "BooleanEquals": false,
        "Next": "No"
      },
      {
        "Variable": "$.IsHelloWorldExample",
        "BooleanEquals": true,
        "Next": "Yes"
      }
    ],
    "Default": "Yes"
  },
  "No": {
    "Type": "Fail",
    "Cause": "Not Hello World"
  },
  "Yes": {
    "Type": "Pass",
    "Next": "Wait 3 sec"
  },
  "Wait 3 sec": {
    "Type": "Wait",
    "Seconds": 3,
    "Next": "Parallel"
  },
  "Parallel": {
    "Type": "Parallel",
    "End": true,
    "Branches": [
      {
        "StartAt": "Hello",
        "States": {
```

```
            "Hello": {
              "Type": "Pass",
              "End": true
            }
          }
        },
        {
          "StartAt": "World",
          "States": {
            "World": {
              "Type": "Pass",
              "End": true
            }
          }
        }
      ]
    }
  }
}
```

Langkah 4: Edit definisi alur kerja dalam mode Kode

Mode Kode Workflow Studio menyediakan editor kode terintegrasi untuk melihat dan mengedit definisi ASL alur kerja Anda.

1. Pilih Kode untuk beralih ke mode Kode.
2. Setelah definisi status Paralel, tempatkan kursor dan tekan**`Enter`**.
3. Tekan **`Ctrl+space`** untuk melihat daftar status yang dapat Anda tambahkan setelah status Paralel.
4. Pilih Pass State dari daftar opsi. Editor kode menambahkan kode boilerplate untuk Pass State.
5. Penambahan status ini menghasilkan kesalahan dalam definisi alur kerja Anda. Dalam definisi keadaan Paralel, ganti `"End": true` dengan**`"Next": "PassState"`**.
6. Dalam definisi Status Lulus yang Anda tambahkan, buat perubahan berikut:
   a. Hapus node Hasil.
   b. Hapus `"ResultPath": "$.result"`, dan`"Next": "NextState"`.
   c. Setelah`"Type": "Pass",`, masuk**`"End": true`**.
   d. Tambahkan definisi **`,`** After Pass State.

Definisi alur kerja Anda sekarang harus terlihat mirip dengan definisi berikut.

```
{
  "Comment": "A description of my state machine",
  "StartAt": "Pass",
  "States": {
    "Pass": {
      "Type": "Pass",
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": true,
          "Next": "Yes"
        },
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": false,
          "Next": "No"
        }
      ],
      "Default": "Yes"
    },
    "Yes": {
      "Type": "Pass",
      "Next": "Wait 3 seconds"
    },
    "Wait 3 seconds": {
      "Type": "Wait",
      "Seconds": 3,
      "Next": "Parallel"
    },
    "Parallel": {
      "Type": "Parallel",
      "Branches": [
        {
          "StartAt": "Hello",
          "States": {
            "Hello": {
              "Type": "Pass",
              "End": true
```

```
            }
          }
        },
        {
          "StartAt": "World",
          "States": {
            "World": {
              "Type": "Pass",
              "End": true
            }
          }
        }
      ],
      "Next": "PassState"
    },
    "PassState": {
      "Type": "Pass",
      "End": true
    },
    "No": {
      "Type": "Fail"
    }
  }
}
```

Langkah 5: Simpan mesin negara

1. Pilih Config more atau pilih ikon edit di sebelah nama mesin status default. MyStateMachine Dalam konfigurasi mesin State, tentukan nama. Misalnya, masukkan **`HelloWorld`**.
2. (Opsional) Tentukan pengaturan alur kerja lainnya, seperti jenis mesin status dan peran pelaksanaannya. Untuk tutorial ini, simpan semua pilihan default dalam konfigurasi mesin State.
3. Pilih Buat.
4. Dalam kotak dialog Konfirmasi pembuatan peran, pilih Konfirmasi untuk melanjutkan.

   Anda juga dapat memilih Lihat konfigurasi peran untuk kembali ke mode Config.

Untuk informasi selengkapnya tentang mode Config, lihat Mode Config Workflow Studio.

Langkah 6: Jalankan mesin negara

Eksekusi mesin status adalah instans tempat Anda menjalankan alur kerja untuk melakukan tugas.

1. Pada halaman mesin negara, pilih mesin HelloWorldnegara.
2. Pada HelloWorldhalaman, pilih Mulai eksekusi.
3. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

   > **Note**
   >
   > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

4. Di kotak Input, masukkan nilai input untuk eksekusi Anda dalam format JSON. Berdasarkan masukan Anda, `IsHelloWorldExample` variabel menentukan aliran mesin status mana yang akan dieksekusi. Untuk saat ini, gunakan nilai input berikut:

   ```
   {
      "IsHelloWorldExample": true
   }
   ```

   > **Note**
   >
   > Sementara menentukan input eksekusi adalah opsional, dalam tutorial ini, adalah wajib untuk menentukan input eksekusi mirip dengan contoh input di atas. Nilai masukan ini direferensikan dalam `Choice` status saat Anda menjalankan mesin status.

5. Pilih Mulai Eksekusi.
6. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

   Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi

masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Untuk tutorial ini, jika Anda memasukkan nilai masukan`"IsHelloWorldExample": true`, Anda akan melihat output berikut:

```
{
   "IsHelloWorldExample": true
},
{
   "IsHelloWorldExample": true
}
```

Langkah 7: Perbarui mesin negara Anda

Ketika Anda memperbarui mesin status, pembaruan Anda pada akhirnya konsisten. Setelah beberapa saat, semua eksekusi yang baru dimulai akan mencerminkan definisi terbaru mesin status Anda. Semua eksekusi yang sedang berjalan akan berjalan hingga selesai di bawah definisi sebelumnya.

Pada langkah ini, Anda akan memperbarui mesin status Anda dalam Mode desain mode Workflow Studio. Anda akan menambahkan `Result` bidang di status Pass bernama World.

1. Pada halaman yang diberi judul dengan ID eksekusi Anda, pilih Edit mesin status.
2. Pastikan Anda berada dalam mode Desain.
3. Pilih status Pass bernama Dunia di kanvas, lalu pilih Output.
4. Di kotak Hasil, masukkan**`"World has been updated!"`**.
5. Pilih Simpan.
6. (Opsional) Di area Definisi, lihat definisi Amazon States Language yang diperbarui dari alur kerja Anda.

```
{
      "Type": "Parallel",
      "End": true,
      "Branches": [
        {
          "StartAt": "Hello",
          "States": {
```

```
            "Hello": {
              "Type": "Pass",
              "End": true
            }
          }
        },
        {
          "StartAt": "World",
          "States": {
            "World": {
              "Type": "Pass",
              "Result": "World has been updated!",
              "End": true
            }
          }
        }
      ],
      "Next": "PassState"
    }
```

7. Pilih Eksekusi.
8. Dalam kotak dialog Mulai eksekusi yang terbuka di tab baru, berikan input eksekusi berikut.

```
{
   "IsHelloWorldExample": true
}
```

9. Pilih Mulai Eksekusi.
10. (Opsional) Dalam tampilan Grafik, pilih langkah Dunia, lalu pilih Keluaran. Outputnya adalah Dunia telah diperbarui!

Langkah 8: Membersihkan

Untuk menghapus mesin status Anda

1. Dari menu navigasi, pilih Mesin status.
2. Pada halaman State Machines, pilih HelloWorld, lalu pilih Delete.
3. Dalam kotak dialog Delete state machine, ketik **`delete`** untuk mengonfirmasi penghapusan.
4. Pilih Hapus.

Jika penghapusan berhasil, bilah status hijau muncul di bagian atas layar Anda. Bilah status hijau memberi tahu Anda bahwa mesin status Anda telah ditandai untuk dihapus. Mesin status Anda akan dihapus ketika semua eksekusi yang sedang berlangsung berhenti berjalan.

Untuk menghapus peran eksekusi

1. Buka Halaman peran untuk IAM.
2. Pilih peran IAM yang dibuat oleh Step Functions untuk Anda. Misalnya, StepFunctions-HelloWorld -Role-Example.
3. Pilih Hapus peran.
4. Pilih Ya, Hapus.

Tutorial untuk Step Functions

Tutorial di bagian ini dapat membantu Anda memahami berbagai aspek yang bekerja dengan AWS Step Functions.

Untuk menyelesaikan tutorial ini, Anda memerlukan AWS akun. Jika Anda tidak memiliki AWS akun, navigasikan ke https://aws.amazon.com/ dan pilih Buat AWS Akun.

Topik

- Membuat mesin status Step Functions yang menggunakan Lambda
- Menangani kondisi kesalahan menggunakan mesin status Step Functions
- Menggunakan status Peta Sebaris untuk mengulangi tindakan
- Menyalin data CSV skala besar menggunakan Peta Terdistribusi
- Memproses seluruh batch data dengan fungsi Lambda
- Memproses item data individual dengan fungsi Lambda
- Memulai Eksekusi Mesin Status dalam Respons terhadap Peristiwa Amazon S3
- Membuat Step Functions API menggunakan API Gateway
- Membuat mesin kondisi Step FunctionsAWS SAM
- Membuat mesin status Aktivitas menggunakan Step Functions
- Ulangi loop dengan Lambda
- Melanjutkan Eksekusi Alur Kerja yang Berjalan Lama sebagai Eksekusi Baru
- Men-deploy Contoh Proyek Persetujuan Manusia
- Lihat jejak X-Ray di Step Functions
- Kumpulkan info bucket Amazon S3 menggunakan integrasi layanan AWS SDK

Membuat mesin status Step Functions yang menggunakan Lambda

Dalam tutorial ini, Anda akan membuat alur kerja satu langkah menggunakan AWS Step Functions untuk memanggil fungsi. AWS Lambda

> **Note**
>
> Step Functions didasarkan pada mesin status dan tugas. Dalam Step Functions, alur kerja disebut state machine, yang merupakan serangkaian langkah yang digerakkan oleh

peristiwa. Setiap langkah dalam alur kerja disebut status. TugasNegara mewakili unit kerja yang dilakukan oleh AWS layanan lain, seperti AWS Lambda,. Status Tugas dapat memanggil API Layanan AWS atau apa pun.
Lihat informasi yang lebih lengkap di:

- Apa itu AWS Step Functions?
- Hubungi AWS layanan lain

Lambda sangat cocok untuk `Task` negara bagian, karena fungsi Lambda tanpa server dan mudah ditulis. Anda dapat menulis kode di AWS Management Console atau editor favorit Anda. AWS menangani detail penyediaan lingkungan komputasi untuk fungsi Anda dan menjalankannya.

Dalam topik ini:

- Langkah 1: Membuat fungsi Lambda
- Langkah 2: Uji fungsi Lambda
- Langkah 3: Buat mesin negara
- Langkah 4: Jalankan mesin negara

Langkah 1: Membuat fungsi Lambda

Fungsi Lambda Anda menerima data peristiwa dan mengembalikan pesan ucapan.

Important

Pastikan fungsi Lambda Anda berada di bawah AWS akun dan AWS Wilayah yang sama dengan mesin negara bagian Anda.

1. Buka konsol Lambda dan pilih Create Function.
2. Pilih halaman Buat fungsi, pilih Penulis dari scratch.
3. Untuk Nama fungsi, masukkan `HelloFunction`.
4. Simpan pilihan default untuk semua opsi lain, lalu pilih Buat fungsi.
5. Setelah fungsi Lambda Anda dibuat, salin Nama Sumber Daya Amazon (ARN) fungsi yang ditampilkan di sudut kanan atas halaman. Untuk menyalin ARN, klik.

Berikut ini adalah contoh ARN:

```
arn:aws:lambda:us-east-1:123456789012:function:HelloFunction
```

6. Salin kode berikut untuk fungsi Lambda ke bagian Sumber kode halaman. ***HelloFunction***

```
export const handler = async(event, context, callback) => {
    callback(null, "Hello from " + event.who + "!");
};
```

Kode ini menyusun sapaan menggunakan bidang `who` data input, yang disediakan oleh objek `event` yang diteruskan ke fungsi Anda. Anda menambahkan data input untuk fungsi ini nanti, ketika Anda memulai eksekusi baru. Metode `callback` mengembalikan sapaan gabungan dari fungsi Anda.

7. Pilih Deploy.

Langkah 2: Uji fungsi Lambda

Uji fungsi Lambda Anda untuk melihatnya beroperasi.

1. Pilih Uji.
2. Untuk Nama peristiwa, masukkan `HelloEvent`.
3. Ganti data Event JSON dengan yang berikut ini.

```
{
    "who": "AWS Step Functions"
}
```

Entri `"who"` sesuai dengan bidang `event.who` di fungsi Lambda Anda, menyelesaikan sapaan. Anda akan memasukkan data input yang sama ketika Anda menjalankan mesin status Anda.

4. Pilih Simpan dan kemudian pilih Uji.
5. Untuk meninjau hasil pengujian, di bawah Hasil eksekusi, perluas Detail.

Langkah 3: Buat mesin negara

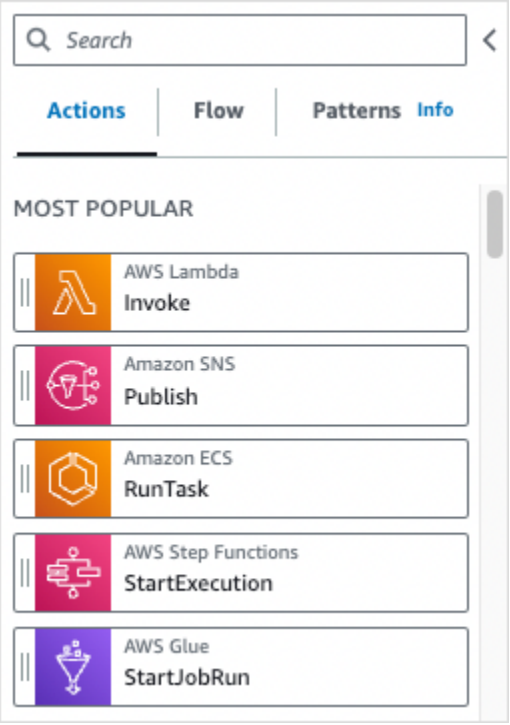
Gunakan konsol Step Functions untuk membuat mesin status yang memanggil fungsi Lambda yang Anda buat di Langkah 1.

1. Buka Konsol Step Functions dan pilih Buat mesin status.

   > **Important**
   >
   > Pastikan mesin status Anda berada di bawah AWS akun dan Wilayah yang sama dengan fungsi Lambda yang Anda buat sebelumnya.

2. Dalam kotak dialog Pilih templat, pilih Kosong.
3. Pilih Pilih. Ini membuka Workflow Studio diMode desain.
4. Di browser States di sebelah kiri, pastikan Anda telah memilih tab Actions. Kemudian, lakukan hal berikut:

   - Seret dan lepas API AWS Lambda Invoke ke status kosong berlabel Seret status pertama di sini.



5. Di panel Inspector di sebelah kanan, konfigurasikan fungsi Lambda:

a. Di bagian Parameter API, pilih fungsi Lambda yang Anda buat sebelumnya di daftar dropdown nama Fungsi.

b. Simpan pilihan default di daftar dropdown Payload.

6. (Opsional) Pilih Definisi untuk melihat definisi state machine Amazon States Language (ASL), yang secara otomatis dihasilkan berdasarkan pilihan Anda di tab Actions dan panel Inspector.

7. Tentukan nama untuk mesin negara Anda. Untuk melakukan ini, pilih ikon edit di sebelah nama mesin status default MyStateMachine. Kemudian, dalam konfigurasi mesin Negara, tentukan nama di kotak Nama mesin Negara.

   Misalnya, masukkan nama**`LambdaStateMachine`**.

   > **Note**
   >
   > Nama mesin negara, eksekusi, dan tugas aktivitas tidak boleh melebihi 80 karakter panjangnya. Nama-nama ini harus unik untuk akun dan AWS Wilayah Anda, dan tidak boleh mengandung salah satu dari yang berikut:
   >
   > - Spasi putih
   > - Karakter wildcard (`? *`)
   > - Karakter tanda kurung (`< > { } [ ]`)
   > - Karakter khusus (`" # % \ ^ | ~ ` $ & , ; : /`)
   > - Karakter kontrol (`\\u0000` - `\\u001f` atau `\\u007f` - `\\u009f`).
   >
   > Jika mesin status Anda bertipe Express, Anda dapat memberikan nama yang sama untuk beberapa eksekusi mesin status. Step Functions menghasilkan ARN eksekusi unik untuk setiap eksekusi mesin status Express, bahkan jika beberapa eksekusi memiliki nama yang sama.
   > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

8. (Opsional) Dalam konfigurasi mesin State, tentukan pengaturan alur kerja lainnya, seperti jenis mesin status dan peran pelaksanaannya.

Untuk tutorial ini, simpan semua pilihan default di pengaturan mesin State.

9. Pilih Buat.
10. Dalam kotak dialog Konfirmasi pembuatan peran, pilih Konfirmasi untuk melanjutkan.

    Anda juga dapat memilih Lihat pengaturan peran untuk kembali ke konfigurasi mesin Status.

> **Note**
>
> Jika Anda menghapus IAM role yang Step Functions buat, Step Functions tidak dapat membuatnya kembali nanti. Demikian pula, jika Anda mengubah peran (misalnya, dengan menghapus Step Functions dari principal dalam kebijakan IAM), Step Functions tidak dapat memulihkan pengaturan aslinya nanti.

Langkah 4: Jalankan mesin negara

Setelah Anda membuat mesin status Anda, Anda dapat menjalankannya.

1. Pada halaman mesin Negara, pilih LambdaStateMachine.
2. Pilih Mulai Eksekusi.

   Kotak dialog Mulai eksekusi ditampilkan.
3. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

> **Note**
>
> Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

4. Di area Input, ganti contoh data eksekusi dengan yang berikut ini.

```
{
    "who" : "AWS Step Functions"
}
```

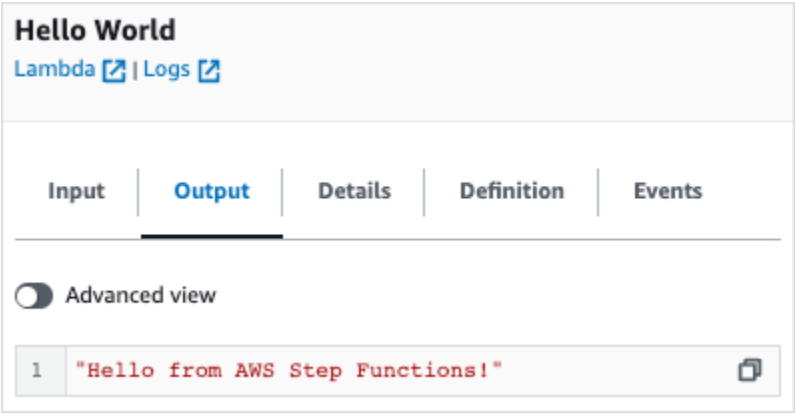
`"who"` adalah nama kunci yang fungsi Lambda Anda gunakan untuk mendapatkan nama orang untuk menyapa.

5. Pilih Mulai Eksekusi.

   Eksekusi state machine Anda dimulai, dan halaman baru yang menunjukkan eksekusi yang sedang berjalan ditampilkan.

6. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

   Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.



> **Note**
>
> Anda juga dapat meneruskan muatan saat memanggil Lambda dari mesin negara. Untuk informasi selengkapnya dan contoh tentang memanggil Lambda dengan meneruskan payload di lapangan, lihat`Parameters`. Panggil Lambda dengan Step Functions

Menangani kondisi kesalahan menggunakan mesin status Step Functions

Dalam tutorial ini, Anda membuat mesin AWS Step Functions negara dengan Status fallback bidang. `Catch`Bidang ini menggunakan AWS Lambda fungsi untuk merespons dengan logika bersyarat berdasarkan jenis pesan kesalahan. Ini adalah teknik yang disebut penanganan kesalahan fungsi.

Untuk informasi selengkapnya, lihat kesalahan AWS Lambda fungsi di Node.js di Panduan AWS Lambda Pengembang.

> **Note**
>
> Anda juga dapat membuat mesin status yang Coba lagi pada batas waktu atau yang digunakan `Catch` untuk transisi ke status tertentu ketika terjadi kesalahan atau batas waktu. Untuk contoh dari teknik penanganan kesalahan ini, lihat Contoh yang Menggunakan Coba Ulang dan Menggunakan Tangkapan.

Dalam topik ini:

- Langkah 1: Buat fungsi Lambda yang gagal
- Langkah 2: Uji fungsi Lambda
- Langkah 3: Buat mesin status dengan bidang Catch
- Langkah 4: Jalankan mesin negara

Langkah 1: Buat fungsi Lambda yang gagal

Gunakan fungsi Lambda untuk menyimulasikan kondisi kesalahan.

> **Important**
>
> Pastikan fungsi Lambda Anda berada di bawah AWS akun dan AWS Wilayah yang sama dengan mesin negara bagian Anda.

1. Buka AWS Lambda konsol di https://console.aws.amazon.com/lambda/.

2. Pilih Buat fungsi.
3. Pilih Gunakan cetak biru, masukkan `step-functions` ke dalam kotak pencarian, lalu pilih Lempar cetak biru kesalahan kustom.
4. Untuk Nama fungsi, masukkan `FailFunction`.
5. Untuk Peran, pertahankan pilihan default (Buat peran baru dengan izin Lambda dasar).
6. Kode berikut ditampilkan dalam panel Kode fungsi Lambda.

```
exports.handler = async (event, context) => {
    function CustomError(message) {
        this.name = 'CustomError';
        this.message = message;
    }
    CustomError.prototype = new Error();

    throw new CustomError('This is a custom error!');
};
```

   Objek `context` mengembalikan pesan kesalahan `This is a custom error!`.
7. Pilih Buat fungsi.
8. Setelah fungsi Lambda Anda dibuat, salin Nama Sumber Daya Amazon (ARN) fungsi yang ditampilkan di sudut kanan atas halaman. Untuk menyalin ARN, klik.

   Berikut ini adalah contoh ARN:

```
arn:aws:lambda:us-east-1:123456789012:function:FailFunction
```

9. Pilih Deploy.

Langkah 2: Uji fungsi Lambda

Uji fungsi Lambda Anda untuk melihatnya beroperasi.

1. Pada FailFunctionhalaman, pilih tab Uji, lalu pilih Uji. Anda tidak perlu membuat acara pengujian.
2. Untuk meninjau hasil pengujian (kesalahan simulasi), di bawah Hasil eksekusi, perluas Detail.

Langkah 3: Buat mesin status dengan bidang Catch

Gunakan konsol Step Functions untuk membuat mesin status yang menggunakan Tugas status dengan `Catch` bidang. Tambahkan referensi ke fungsi Lambda Anda dalam status Tugas. Mesin status memanggil fungsi Lambda, yang gagal selama eksekusi. Step Functions mencoba ulang fungsi dua kali menggunakan backoff eksponensial di antara beberapa percobaan ulang.

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Dalam kotak dialog Pilih templat, pilih Kosong.
3. Pilih Pilih. Ini membuka Workflow Studio diMode desain.
4. Pilih Kode untuk membuka editor kode. Di editor kode, Anda menulis dan mengedit definisi Amazon States Language (ASL) alur kerja Anda.
5. Tempel kode berikut, tetapi ganti ARN dari fungsi Lambda yang Anda buat sebelumnya di bidang. `Resource`

```
{
   "Comment": "A Catch example of the Amazon States Language using an AWS Lambda
 function",
   "StartAt": "CreateAccount",
   "States": {
      "CreateAccount": {
         "Type": "Task",
         "Resource": "arn:aws:lambda:us-east-1:123456789012:function:FailFunction",
         "Catch": [ {
            "ErrorEquals": ["CustomError"],
            "Next": "CustomErrorFallback"
         }, {
            "ErrorEquals": ["States.TaskFailed"],
            "Next": "ReservedTypeFallback"
         }, {
            "ErrorEquals": ["States.ALL"],
            "Next": "CatchAllFallback"
         } ],
         "End": true
      },
      "CustomErrorFallback": {
         "Type": "Pass",
         "Result": "This is a fallback from a custom Lambda function exception",
         "End": true
      },
```

```
      "ReservedTypeFallback": {
         "Type": "Pass",
         "Result": "This is a fallback from a reserved error code",
         "End": true
      },
      "CatchAllFallback": {
         "Type": "Pass",
         "Result": "This is a fallback from any error code",
         "End": true
      }
   }
}
```

ARN ini adalah deskripsi dari mesin status Anda menggunakan Bahasan Status Amazon. ARN menentukan satu status `Task` bernama `CreateAccount`. Untuk informasi selengkapnya, lihat Struktur Mesin Status.

Untuk informasi selengkapnya tentang sintaksis bidang `Retry`, lihat Nyatakan contoh mesin menggunakan Coba Ulang dan menggunakan Catch.

> **Note**
>
> Kesalahan yang tidak tertangani di Lambda dilaporkan sebagai `Lambda.Unknown` dalam output kesalahan. Ini termasuk out-of-memory kesalahan dan batas waktu fungsi. Anda dapat mencocokkan di `Lambda.Unknown`, `States.ALL`, atau `States.TaskFailed` untuk menangani kesalahan ini. Ketika Lambda mencapai jumlah maksimum permintaan, kesalahannya adalah `Lambda.TooManyRequestsException`. Untuk informasi selengkapnya tentang kesalahan fungsi Lambda, lihat Penanganan kesalahan dan percobaan ulang otomatis di Panduan PengembangAWS Lambda .

6. (Opsional) DiPanel visualisasi grafik, lihat visualisasi grafis real-time dari alur kerja Anda.
7. Tentukan nama untuk mesin negara Anda. Untuk melakukan ini, pilih ikon edit di sebelah nama mesin status default MyStateMachine. Kemudian, dalam konfigurasi mesin Negara, tentukan nama di kotak Nama mesin Negara.

   Untuk tutorial ini, masukkan **`Catchfailure`**.
8. (Opsional) Dalam konfigurasi mesin State, tentukan pengaturan alur kerja lainnya, seperti jenis mesin status dan peran pelaksanaannya.

Untuk tutorial ini, simpan semua pilihan default di pengaturan mesin State.

9. Dalam kotak dialog Konfirmasi pembuatan peran, pilih Konfirmasi untuk melanjutkan.

   Anda juga dapat memilih Lihat pengaturan peran untuk kembali ke konfigurasi mesin Status.

   > **Note**
   >
   > Jika Anda menghapus IAM role yang Step Functions buat, Step Functions tidak dapat membuatnya kembali nanti. Demikian pula, jika Anda mengubah peran (misalnya, dengan menghapus Step Functions dari principal dalam kebijakan IAM), Step Functions tidak dapat memulihkan pengaturan aslinya nanti.

Langkah 4: Jalankan mesin negara

Setelah Anda membuat mesin status Anda, Anda dapat menjalankannya.

1. Pada halaman State Machines, pilih Catchfailure.
2. Pada halaman Catchfailure, pilih Mulai eksekusi. Kotak dialog Mulai eksekusi ditampilkan.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

   1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

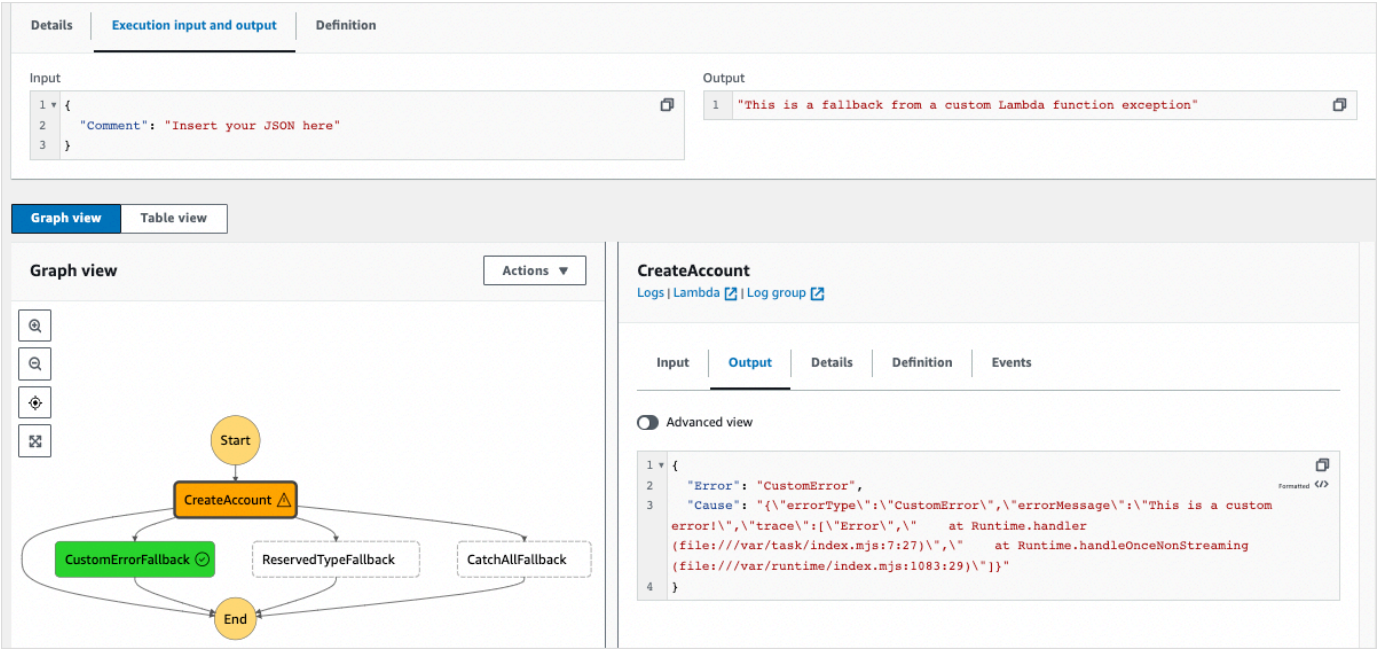
      > **Note**
      >
      > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

   2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.
   3. Pilih Mulai Eksekusi.

4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

   Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Misalnya, untuk melihat pesan kesalahan kustom Anda, pilih CreateAccountlangkah dalam tampilan Grafik, lalu pilih Output tab.



> **Note**
>
> Anda dapat mempertahankan input status dengan kesalahan dengan menggunakan `ResultPath`. Lihat Gunakan ResultPath untuk Menyertakan Kesalahan dan Input dalam `Catch`.

Menggunakan status Peta Sebaris untuk mengulangi tindakan

Tutorial ini membantu Anda memulai dengan menggunakan `Map` status dalam mode Inline. Anda menggunakan status Peta Sebaris dalam alur kerja untuk melakukan tindakan berulang kali. Untuk informasi selengkapnya tentang mode Inline, lihat Status peta dalam mode Inline.

Dalam tutorial ini, Anda menggunakan status Inline Map untuk berulang kali menghasilkan versi 4 pengidentifikasi unik universal (v4 UUID). Anda mulai dengan membuat alur kerja yang berisi dua Diteruskan status dan status Peta Inline di Workflow Studio. Kemudian, Anda mengonfigurasi input dan output, termasuk array JSON input untuk `Map` status. `Map`Status mengembalikan array output yang berisi UUID v4 yang dihasilkan untuk setiap item dalam array input.

Daftar Isi

- Langkah 1: Buat prototipe alur kerja
- Langkah 2: Konfigurasikan input dan output
- Langkah 3: Tinjau definisi Bahasa Amazon States yang dibuat secara otomatis dan simpan alur kerjanya
- Langkah 4: Jalankan mesin negara

Langkah 1: Buat prototipe alur kerja

Pada langkah ini, Anda membuat prototipe untuk alur kerja Anda menggunakan Workflow Studio. Workflow Studio adalah desainer alur kerja visual yang tersedia di konsol Step Functions. Anda akan memilih status yang diperlukan dari tab Flow dan menggunakan fitur drag and drop Workflow Studio untuk membuat prototipe alur kerja.

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Dalam kotak dialog Pilih templat, pilih Kosong.
3. Pilih Pilih. Ini membuka Workflow Studio diMode desain.
4. Dari tab Flow, seret status Pass dan jatuhkan ke status kosong berlabel Drag first state di sini.
5. Seret status Peta dan jatuhkan di bawah status Pass. Ubah nama status Peta menjadi**`Map demo`**.
6. Seret status Pass kedua dan letakkan di dalam status demo Peta.
7. Ubah nama status Pass kedua menjadi**`Generate UUID`**.

Langkah 2: Konfigurasikan input dan output

Pada langkah ini, Anda mengonfigurasi input dan output untuk semua status dalam prototipe alur kerja Anda. Pertama, Anda menyuntikkan beberapa data tetap ke dalam alur kerja menggunakan status Pass pertama. Status Pass ini meneruskan data ini sebagai masukan ke status demo Map. Dalam input ini, Anda menentukan node yang berisi array input yang harus diulang oleh status demo Peta. Kemudian Anda menentukan langkah yang harus diulang oleh status demo Peta untuk menghasilkan UUID v4. Akhirnya, Anda mengonfigurasi output untuk kembali untuk setiap pengulangan.

1. Pilih status Pass pertama dalam prototipe alur kerja Anda. Di tab Output, masukkan yang berikut ini di bawah Hasil:

```
{
  "foo": "bar",
  "colors": [
    "red",
    "green",
    "blue",
    "yellow",
    "white"
  ]
}
```

2. Pilih status demo Peta dan di tab Konfigurasi, lakukan hal berikut:

   a. Pilih Menyediakan jalur ke array item.

   b. Tentukan jalur referensi berikut untuk memilih node yang berisi array input:

```
$.colors
```

3. Pilih status Generate UUID dan di tab Input, lakukan hal berikut:

   a. Pilih Transform input dengan Parameter.

   b. Masukkan input JSON berikut untuk menghasilkan UUID v4 untuk setiap item array input. Anda menggunakan fungsi `States.UUID` intrinsik untuk menghasilkan UUID.

```
{
  "uuid.$": "States.UUID()"
}
```

4. Untuk status Generate UUID, pilih tab Output dan lakukan hal berikut:

   a. Pilih Filter output dengan OutputPath.

   b. Masukkan jalur referensi berikut untuk memilih node JSON yang berisi item array keluaran:

```
$.uuid
```

Langkah 3: Tinjau definisi Bahasa Amazon States yang dibuat secara otomatis dan simpan alur kerjanya

Saat Anda menyeret dan melepas status dari panel Flow ke kanvas, Workflow Studio secara otomatis menyusun definisi Amazon States Language (ASL) dari alur kerja Anda secara real-time. Anda dapat mengedit definisi ini sesuai kebutuhan.

1. (Opsional) Pilih Definisi pada Inspector panel untuk melihat definisi Amazon States Language yang dihasilkan secara otomatis dari alur kerja Anda.

> **Tip**
>
> Anda juga dapat melihat definisi ASL di Editor kode Workflow Studio. Di editor kode, Anda juga dapat mengedit definisi ASL dari alur kerja Anda.

   Contoh berikut menunjukkan definisi Bahasa Amazon States yang dibuat secara otomatis untuk alur kerja Anda.

```
{
    "Comment": "Using Map state in Inline mode",
    "StartAt": "Pass",
    "States": {
      "Pass": {
        "Type": "Pass",
        "Next": "Map demo",
        "Result": {
          "foo": "bar",
          "colors": [
            "red",
            "green",
            "blue",
```

```
            "yellow",
            "white"
          ]
        }
      },
      "Map demo": {
        "Type": "Map",
        "ItemsPath": "$.colors",
        "ItemProcessor": {
          "ProcessorConfig": {
            "Mode": "INLINE"
          },
          "StartAt": "Generate UUID",
          "States": {
            "Generate UUID": {
              "Type": "Pass",
              "End": true,
              "Parameters": {
                "uuid.$": "States.UUID()"
              },
              "OutputPath": "$.uuid"
            }
          }
        },
        "End": true
      }
    }
  }
```

2. Tentukan nama untuk mesin negara Anda. Untuk melakukan ini, pilih ikon edit di sebelah nama mesin status default MyStateMachine. Kemudian, dalam konfigurasi mesin Negara, tentukan nama di kotak Nama mesin Negara.

   Untuk tutorial ini, masukkan nama **`InlineMapDemo`**.

3. (Opsional) Dalam konfigurasi mesin State, tentukan pengaturan alur kerja lainnya, seperti jenis mesin status dan peran pelaksanaannya.

   Untuk tutorial ini, simpan semua pilihan default dalam konfigurasi mesin State.

4. Dalam kotak dialog Konfirmasi pembuatan peran, pilih Konfirmasi untuk melanjutkan.

   Anda juga dapat memilih Lihat pengaturan peran untuk kembali ke konfigurasi mesin Status.

> **Note**
>
> Jika Anda menghapus IAM role yang Step Functions buat, Step Functions tidak dapat membuatnya kembali nanti. Demikian pula, jika Anda mengubah peran (misalnya, dengan menghapus Step Functions dari principal dalam kebijakan IAM), Step Functions tidak dapat memulihkan pengaturan aslinya nanti.

Langkah 4: Jalankan mesin negara

Eksekusi mesin status adalah instans tempat Anda menjalankan alur kerja untuk melakukan tugas.

1. Pada InlineMapDemohalaman, pilih Mulai eksekusi.
2. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

    1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions secara otomatis menghasilkan nama eksekusi yang unik.

        > **Note**
        >
        > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, aktivitas, dan label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

    2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.
    3. Pilih Mulai Eksekusi.
    4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

        Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Untuk melihat input dan output eksekusi side-by-side, pilih Input dan output eksekusi. Di bawah Output, lihat array output yang dikembalikan oleh `Map` negara. Berikut ini adalah contoh dari array output:

```
[
  "a85cbc7b-4e65-4ac2-97af-80ed504adc1d",
  "b05bca11-d481-414e-aa9a-88285ec6590d",
  "f42d59f7-bd32-480f-b270-caddb518ce2a",
  "15f18616-517d-4b69-b7c3-bf22222d2efd",
  "690bcfee-6d58-408c-a6b4-1995ccafdbd2"
]
```

Menyalin data CSV skala besar menggunakan Peta Terdistribusi

Tutorial ini membantu Anda mulai menggunakan `Map` status dalam mode Terdistribusi. `MapStatus` yang disetel ke Distributed dikenal sebagai status Peta Terdistribusi. Anda menggunakan status Peta Terdistribusi dalam alur kerja Anda untuk mengulangi sumber data Amazon S3 skala besar. `MapStatus` menjalankan setiap iterasi sebagai eksekusi alur kerja anak, yang memungkinkan konkurensi tinggi. Untuk informasi selengkapnya tentang mode Terdistribusi, lihat Status peta dalam mode Terdistribusi.

Dalam tutorial ini, Anda menggunakan status Peta Terdistribusi untuk mengulangi file CSV di bucket Amazon S3. Anda kemudian mengembalikan isinya, bersama dengan ARN eksekusi alur kerja anak, di bucket Amazon S3 lainnya. Anda mulai dengan membuat prototipe alur kerja di Workflow Studio. Selanjutnya, Anda mengatur mode pemrosesan `Map` status ke Distributed, tentukan file CSV sebagai kumpulan data, dan berikan lokasinya ke status. `Map` Anda juga menentukan jenis alur kerja untuk eksekusi alur kerja anak yang status Peta Terdistribusi dimulai sebagai Express.

Selain pengaturan ini, Anda juga menentukan konfigurasi lain, seperti jumlah maksimum eksekusi alur kerja anak bersamaan dan lokasi untuk mengekspor `Map` hasilnya, untuk contoh alur kerja yang digunakan dalam tutorial ini.

Daftar Isi

- Prasyarat
- Langkah 1: Buat prototipe alur kerja
- Langkah 2: Konfigurasikan bidang yang diperlukan untuk status Peta

- Langkah 3: Konfigurasikan opsi tambahan
- Langkah 4: Konfigurasikan fungsi Lambda
- Langkah 5: Perbarui prototipe alur kerja
- Langkah 6: Tinjau definisi Bahasa Amazon States yang dibuat secara otomatis dan simpan alur kerjanya
- Langkah 7: Jalankan mesin negara

Prasyarat

- Unggah file CSV ke bucket Amazon S3. Anda harus menentukan baris header dalam file CSV Anda. Untuk informasi tentang batas ukuran yang dikenakan pada file CSV dan cara menentukan baris header, lihatFile CSV dalam ember Amazon S3.
- Buat bucket Amazon S3 lain dan folder di dalam bucket itu untuk mengekspor hasil `Map` status ke.

> **Important**
>
> Pastikan bucket Amazon S3 Anda berada di bawah yang sama Akun AWS dan Wilayah AWS sebagai mesin negara bagian Anda.

Langkah 1: Buat prototipe alur kerja

Pada langkah ini, Anda membuat prototipe untuk alur kerja Anda menggunakan Workflow Studio. Workflow Studio adalah desainer alur kerja visual yang tersedia di konsol Step Functions. Anda memilih status dan tindakan API yang diperlukan dari tab Flow dan Actions masing-masing. Anda akan menggunakan fitur drag and drop Workflow Studio untuk membuat prototipe alur kerja.

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Dalam kotak dialog Pilih templat, pilih Kosong.
3. Pilih Pilih. Ini membuka Workflow Studio diMode desain.
4. Dari tab Flow, seret status Peta dan jatuhkan ke status kosong berlabel Seret status pertama di sini.
5. Di tab Konfigurasi, untuk nama Negara, masukkan**`Process data`**.

6. Dari tab Tindakan, seret tindakan AWS Lambda Invoke API dan letakkan di dalam status data Proses.
7. Ubah nama status AWS Lambda Invoke menjadi. **`Process CSV data`**

Langkah 2: Konfigurasikan bidang yang diperlukan untuk status Peta

Pada langkah ini, Anda mengonfigurasi bidang wajib berikut dari status Peta Terdistribusi:

- ItemReader— Menentukan dataset dan lokasi dari mana `Map` negara dapat membaca input.
- ItemProcessor- Menentukan nilai-nilai berikut:
  - `ProcessorConfig`— Mengatur `Mode` dan `ExecutionType` ke `DISTRIBUTED` dan `EXPRESS` masing-masing. Ini menetapkan mode pemrosesan `Map` status dan jenis alur kerja untuk eksekusi alur kerja anak yang memulai status Peta Terdistribusi.
  - `StartAt`— Status pertama dalam alur kerja Peta.
  - `States`— Mendefinisikan alur kerja Peta, yang merupakan serangkaian langkah untuk diulang dalam setiap eksekusi alur kerja anak.
- ResultWriter— Menentukan lokasi Amazon S3 tempat Step Functions menulis hasil status Peta Terdistribusi.

> **Important**
>
> Pastikan bucket Amazon S3 yang Anda gunakan untuk mengekspor hasil Map Run berada di bawah yang sama Akun AWS dan Wilayah AWS sebagai mesin status Anda. Jika tidak, eksekusi mesin status Anda akan gagal dengan `States.ResultWriterFailed` kesalahan tersebut.

Untuk mengkonfigurasi bidang yang diperlukan:

1. Pilih status data Proses dan, di tab Konfigurasi, lakukan hal berikut:
   a. Untuk mode Pemrosesan, pilih Didistribusikan.
   b. Untuk Sumber item, pilih Amazon S3, lalu pilih file CSV di S3 dari daftar tarik-turun sumber item S3.
   c. Lakukan hal berikut untuk menentukan lokasi Amazon S3 file CSV Anda:

i. Untuk objek S3, pilih Masukkan bucket dan kunci dari daftar dropdown.

ii. Untuk Bucket, masukkan nama bucket Amazon S3, yang berisi file CSV. Misalnya, **`sourceBucket`**.

iii. Untuk Kunci, masukkan nama objek Amazon S3 tempat Anda menyimpan file CSV. Anda juga harus menentukan nama file CSV di bidang ini. Misalnya, **`csvDataset/ratings.csv`**.

d. Untuk file CSV, Anda juga harus menentukan lokasi header kolom. Untuk melakukan ini, pilih Konfigurasi tambahan, lalu untuk lokasi header CSV pertahankan pilihan default Baris pertama jika baris pertama file CSV Anda adalah header. Jika tidak, pilih Diberikan untuk menentukan header dalam definisi mesin negara. Untuk informasi selengkapnya, lihat `ReaderConfig`.

e. Untuk jenis eksekusi Anak, pilih Express.

2. Di lokasi Ekspor, untuk mengekspor hasil Map Run ke lokasi Amazon S3 tertentu, pilih Ekspor output status Peta ke Amazon S3.

3. Lakukan hal-hal berikut:

a. Untuk bucket S3, pilih Masukkan nama bucket dan awalan dari daftar dropdown.

b. Untuk Bucket, masukkan nama bucket Amazon S3 tempat Anda ingin mengekspor hasilnya. Misalnya, **`mapOutputs`**.

c. Untuk Awalan, masukkan nama folder tempat Anda ingin menyimpan hasilnya. Misalnya, **`resultData`**.

Langkah 3: Konfigurasikan opsi tambahan

Selain pengaturan yang diperlukan untuk status Peta Terdistribusi, Anda juga dapat menentukan opsi lain. Ini dapat mencakup jumlah maksimum eksekusi alur kerja anak bersamaan dan lokasi untuk mengekspor hasil `Map` status.

1. Pilih status data Proses. Kemudian, di Sumber item, pilih Konfigurasi tambahan.

2. Lakukan hal-hal berikut:

a. Pilih Ubah item dengan ItemSelector untuk menentukan input JSON khusus untuk setiap eksekusi alur kerja anak.

b. Masukkan input JSON berikut:

```
{
  "index.$": "$$.Map.Item.Index",
  "value.$": "$$.Map.Item.Value"
}
```

Untuk informasi tentang cara membuat input kustom, lihat`ItemSelector`.

3. Dalam pengaturan Runtime, untuk batas Konkurensi, tentukan jumlah eksekusi alur kerja turunan bersamaan yang dapat dimulai oleh status Peta Terdistribusi. Misalnya, masukkan **`100`**.
4. Buka jendela atau tab baru di browser Anda dan selesaikan konfigurasi fungsi Lambda yang akan Anda gunakan dalam alur kerja ini, seperti yang dijelaskan di. Langkah 4: Konfigurasikan fungsi Lambda

Langkah 4: Konfigurasikan fungsi Lambda

> **Important**
>
> Pastikan fungsi Lambda Anda Wilayah AWS sama dengan mesin status Anda.

1. Buka konsol Lambda dan pilih Buat fungsi.
2. Pilih halaman Buat fungsi, pilih Penulis dari scratch.
3. Di bagian Informasi dasar, konfigurasikan fungsi Lambda Anda:
   a. Untuk Nama fungsi, masukkan **`distributedMapLambda`**.
   b. Untuk Runtime, pilih Node.js 16.x.
   c. Simpan semua pilihan default dan pilih Create function.
   d. Setelah Anda membuat fungsi Lambda, salin Nama Sumber Daya Amazon (ARN) fungsi yang ditampilkan di sudut kanan atas halaman. Anda harus menyediakan ini dalam prototipe alur kerja Anda. Untuk menyalin ARN, klik.

   Berikut ini adalah contoh ARN:

```
arn:aws:lambda:us-east-2:123456789012:function:distributedMapLambda
```

4. Salin kode berikut untuk fungsi Lambda dan tempel ke bagian Sumber kode halaman. distributedMapLambda

```
exports.handler = async function(event, context) {
    console.log("Received Input:\n", event);

    return {
        'statusCode' : 200,
        'inputReceived' : event //returns the input that it received
    }
};
```

5. Pilih Deploy. Setelah fungsi Anda di-deploy, pilih Test untuk melihat output dari fungsi Lambda Anda.

Langkah 5: Perbarui prototipe alur kerja

Di konsol Step Functions, Anda akan memperbarui alur kerja Anda untuk menambahkan ARN fungsi Lambda.

1. Kembali ke tab atau jendela tempat Anda membuat prototipe alur kerja.
2. Pilih langkah data Proses CSV, dan di tab Konfigurasi, lakukan hal berikut:
   a. Untuk jenis Integrasi, pilih Dioptimalkan.
   b. Untuk nama Fungsi, mulailah memasukkan nama fungsi Lambda Anda. Pilih fungsi dari daftar dropdown yang muncul, atau pilih Masukkan nama fungsi dan berikan ARN fungsi Lambda.

Langkah 6: Tinjau definisi Bahasa Amazon States yang dibuat secara otomatis dan simpan alur kerjanya

Saat Anda menyeret dan melepas status dari tab Action dan Flow ke kanvas, Workflow Studio secara otomatis menyusun definisi Amazon States Language dari alur kerja Anda secara real-time. Anda dapat mengedit definisi ini sesuai kebutuhan.

1. (Opsional) Pilih Definisi pada Inspector panel dan lihat definisi mesin status.

> **Tip**
>
> Anda juga dapat melihat definisi ASL di Editor kode Workflow Studio. Di editor kode, Anda juga dapat mengedit definisi ASL dari alur kerja Anda.

Kode contoh berikut menunjukkan definisi Bahasa Negara Amazon yang dibuat secara otomatis untuk alur kerja Anda.

```
{
  "Comment": "Using Map state in Distributed mode",
  "StartAt": "Process data",
  "States": {
    "Process data": {
      "Type": "Map",
      "MaxConcurrency": 100,
      "ItemReader": {
        "ReaderConfig": {
          "InputType": "CSV",
          "CSVHeaderLocation": "FIRST_ROW"
        },
        "Resource": "arn:aws:states:::s3:getObject",
        "Parameters": {
          "Bucket": "sourceBucket",
          "Key": "csvDataset/ratings.csv"
        }
      },
      "ItemProcessor": {
        "ProcessorConfig": {
          "Mode": "DISTRIBUTED",
          "ExecutionType": "EXPRESS"
        },
        "StartAt": "Process CSV data",
        "States": {
          "Process CSV data": {
            "Type": "Task",
            "Resource": "arn:aws:states:::lambda:invoke",
            "OutputPath": "$.Payload",
            "Parameters": {
              "Payload.$": "$",
```

```
              "FunctionName": "arn:aws:lambda:us-
east-2:123456789012:function:distributedMapLambda"
            },
            "End": true
          }
        }
      },
      "Label": "Processdata",
      "End": true,
      "ResultWriter": {
        "Resource": "arn:aws:states:::s3:putObject",
        "Parameters": {
          "Bucket": "mapOutputs",
          "Prefix": "resultData"
        }
      },
      "ItemSelector": {
        "index.$": "$$.Map.Item.Index",
        "value.$": "$$.Map.Item.Value"
      }
    }
  }
}
```

2. Tentukan nama untuk mesin negara Anda. Untuk melakukan ini, pilih ikon edit di sebelah nama mesin status default MyStateMachine. Kemudian, dalam konfigurasi mesin Negara, tentukan nama di kotak Nama mesin Negara.

   Untuk tutorial ini, masukkan nama **`DistributedMapDemo`**.

3. (Opsional) Dalam konfigurasi mesin State, tentukan pengaturan alur kerja lainnya, seperti jenis mesin status dan peran pelaksanaannya.

   Untuk tutorial ini, simpan semua pilihan default dalam konfigurasi mesin State.

4. Dalam kotak dialog Konfirmasi pembuatan peran, pilih Konfirmasi untuk melanjutkan.

   Anda juga dapat memilih Lihat pengaturan peran untuk kembali ke konfigurasi mesin Status.

> **Note**
>
> Jika Anda menghapus IAM role yang Step Functions buat, Step Functions tidak dapat membuatnya kembali nanti. Demikian pula, jika Anda mengubah peran (misalnya,

dengan menghapus Step Functions dari principal dalam kebijakan IAM), Step Functions tidak dapat memulihkan pengaturan aslinya nanti.

Langkah 7: Jalankan mesin negara

Eksekusi adalah instance dari mesin status tempat Anda menjalankan alur kerja untuk melakukan tugas.

1. Pada DistributedMapDemohalaman, pilih Mulai eksekusi.
2. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

    1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions secara otomatis menghasilkan nama eksekusi yang unik.

        > **Note**
        >
        > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, aktivitas, dan label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

    2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.
    3. Pilih Mulai Eksekusi.
    4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

        Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Misalnya, pilih `Map` status, lalu pilih Map Run untuk membuka halaman Map Run Details. Di halaman ini, Anda dapat melihat semua detail eksekusi status Peta Terdistribusi dan eksekusi alur kerja anak yang dimulai. Untuk informasi tentang halaman ini, lihatMemeriksa Map Run.

Memproses seluruh batch data dengan fungsi Lambda

Dalam tutorial ini, Anda menggunakan ItemBatcher bidang status Peta Terdistribusi untuk memproses seluruh kumpulan item di dalam fungsi Lambda. Setiap batch berisi maksimal tiga item. Status Peta Terdistribusi memulai empat eksekusi alur kerja anak, di mana setiap eksekusi memproses tiga item, sementara satu eksekusi memproses satu item. Setiap eksekusi alur kerja anak memanggil fungsi Lambda yang mengulang item individual yang ada dalam batch.

Anda akan membuat mesin status yang melakukan perkalian pada array bilangan bulat. Katakanlah bahwa array integer yang Anda berikan sebagai input adalah `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]` dan faktor perkalian adalah. 7 Kemudian, array yang dihasilkan terbentuk setelah mengalikan bilangan bulat ini dengan faktor 7, akan menjadi. `[7, 14, 21, 28, 35, 42, 49, 56, 63, 70]`

Topik

- Langkah 1: Buat mesin negara
- Langkah 2: Buat fungsi Lambda
- Langkah 3: Jalankan mesin negara

Langkah 1: Buat mesin negara

Pada langkah ini, Anda membuat prototipe alur kerja mesin status yang meneruskan seluruh kumpulan data ke fungsi Lambda yang akan Anda buat di Langkah 2.

- Gunakan definisi berikut untuk membuat mesin status menggunakan konsol Step Functions. Untuk informasi tentang membuat mesin status, lihat Langkah 1: Buat prototipe alur kerja di Memulai dengan menggunakan Distributed Map state tutorial.

  Dalam mesin status ini, Anda mendefinisikan status Peta Terdistribusi yang menerima array 10 bilangan bulat sebagai input dan meneruskan array ini ke fungsi Lambda dalam batch. 3 Fungsi Lambda mengulangi item individual yang ada dalam batch dan mengembalikan array keluaran

bernama. `multiplied` Array keluaran berisi hasil perkalian yang dilakukan pada item yang dilewatkan dalam array input.

> **Important**
>
> Pastikan untuk mengganti Amazon Resource Name (ARN) dari fungsi Lambda dalam kode berikut dengan ARN dari fungsi yang akan Anda buat di Langkah 2.

```
{
  "StartAt": "Pass",
  "States": {
    "Pass": {
      "Type": "Pass",
      "Next": "Map",
      "Result": {
        "MyMultiplicationFactor": 7,
        "MyItems": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
      }
    },
    "Map": {
      "Type": "Map",
      "ItemProcessor": {
        "ProcessorConfig": {
          "Mode": "DISTRIBUTED",
          "ExecutionType": "STANDARD"
        },
        "StartAt": "Lambda Invoke",
        "States": {
          "Lambda Invoke": {
            "Type": "Task",
            "Resource": "arn:aws:states:::lambda:invoke",
            "OutputPath": "$.Payload",
            "Parameters": {
              "Payload.$": "$",
              "FunctionName": "arn:aws:lambda:us-
east-1:123456789012:function:functionName"
            },
            "Retry": [
              {
                "ErrorEquals": [
                  "Lambda.ServiceException",
```

```
                  "Lambda.AWSLambdaException",
                  "Lambda.SdkClientException",
                  "Lambda.TooManyRequestsException"
                ],
                "IntervalSeconds": 2,
                "MaxAttempts": 6,
                "BackoffRate": 2
              }
            ],
            "End": true
          }
        }
      },
      "End": true,
      "Label": "Map",
      "MaxConcurrency": 1000,
      "ItemBatcher": {
        "MaxItemsPerBatch": 3,
        "BatchInput": {
          "MyMultiplicationFactor.$": "$.MyMultiplicationFactor"
        }
      },
      "ItemsPath": "$.MyItems"
    }
  }
}
```

Langkah 2: Buat fungsi Lambda

Pada langkah ini, Anda membuat fungsi Lambda yang memproses semua item yang diteruskan dalam batch.

> **Important**
>
> Pastikan fungsi Lambda Anda Wilayah AWS sama dengan mesin status Anda.

Untuk membuat fungsi Lambda

1. Gunakan konsol Lambda untuk membuat nama fungsi Lambda Python 3.9. **`ProcessEntireBatch`** Untuk informasi tentang membuat fungsi Lambda, lihat Langkah 4:

Mengkonfigurasi fungsi Lambda dalam Memulai dengan menggunakan Distributed Map state tutorial.

2. Salin kode berikut untuk fungsi Lambda dan tempelkan ke bagian sumber Kode fungsi Lambda Anda.

```
import json

def lambda_handler(event, context):
    multiplication_factor = event['BatchInput']['MyMultiplicationFactor']
    items = event['Items']

    results = [multiplication_factor * item for item in items]

    return {
        'statusCode': 200,
        'multiplied': results
    }
```

3. Setelah Anda membuat fungsi Lambda Anda, salin ARN fungsi yang ditampilkan di sudut kanan atas halaman. Untuk menyalin ARN, klik tombol.

Berikut ini adalah contoh ARN, di mana *function-name*adalah nama fungsi Lambda (dalam hal ini,): `ProcessEntireBatch`

```
arn:aws:lambda:us-east-1:123456789012:function:function-name
```

Anda harus menyediakan fungsi ARN di mesin status yang Anda buat di Langkah 1.

4. Pilih Terapkan untuk menyebarkan perubahan.

Langkah 3: Jalankan mesin negara

Saat Anda menjalankan mesin status, status Peta Terdistribusi memulai empat eksekusi alur kerja anak, di mana setiap eksekusi memproses tiga item, sementara satu eksekusi memproses satu item.

Contoh berikut menunjukkan data yang diteruskan ke `ProcessEntireBatch`fungsi oleh salah satu eksekusi alur kerja anak.

```
{
  "BatchInput": {
```

```
    "MyMultiplicationFactor": 7
  },
  "Items": [1, 2, 3]
}
```

Dengan masukan ini, contoh berikut menunjukkan array keluaran bernama `multiplied` yang dikembalikan oleh fungsi Lambda.

```
{
  "statusCode": 200,
  "multiplied": [7, 14, 21]
}
```

Mesin state mengembalikan output berikut yang berisi empat array yang dinamai `multiplied` untuk empat eksekusi alur kerja anak. Array ini berisi hasil perkalian dari masing-masing item input.

```
[
  {
    "statusCode": 200,
    "multiplied": [7, 14, 21]
  },
  {
    "statusCode": 200,
    "multiplied": [28, 35, 42]
  },
  {
    "statusCode": 200,
    "multiplied": [49, 56, 63]
  },
  {
    "statusCode": 200,
    "multiplied": [70]
  }
]
```

Untuk menggabungkan semua item array yang dikembalikan ke dalam array keluaran tunggal, Anda dapat menggunakan ResultSelector bidang tersebut. Tentukan bidang ini di dalam status Peta Terdistribusi untuk menemukan semua `multiplied` array, ekstrak semua item di dalam array ini, dan kemudian menggabungkan mereka ke dalam array output tunggal.

Untuk menggunakan `ResultSelector` bidang, perbarui definisi mesin status Anda seperti yang ditunjukkan pada contoh berikut.

```
{
  "StartAt": "Pass",
  "States": {
    ...
    ...
    "Map": {
      "Type": "Map",
      ...
      ...
      "ItemsPath": "$.MyItems",
      "ResultSelector": {
        "multiplied.$": "$..multiplied[*]"
      }
    }
  }
}
```

Mesin status diperbarui mengembalikan array output konsolidasi seperti yang ditunjukkan pada contoh berikut.

```
{
  "multiplied": [7, 14, 21, 28, 35, 42, 49, 56, 63, 70]
}
```

Memproses item data individual dengan fungsi Lambda

Dalam tutorial ini, Anda menggunakan ItemBatcher bidang status Peta Terdistribusi untuk mengulangi item individual yang ada dalam batch menggunakan fungsi Lambda. Status Peta Terdistribusi memulai empat eksekusi alur kerja anak. Masing-masing alur kerja anak ini menjalankan status Peta Inline. Untuk setiap iterasinya, status Peta Inline memanggil fungsi Lambda dan meneruskan satu item dari batch ke fungsi. Fungsi Lambda kemudian memproses item dan mengembalikan hasilnya.

Anda akan membuat mesin status yang melakukan perkalian pada array bilangan bulat. Katakanlah bahwa array integer yang Anda berikan sebagai input adalah `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]` dan faktor perkalian adalah. 7 Kemudian, array yang dihasilkan terbentuk setelah mengalikan

bilangan bulat ini dengan faktor 7, akan menjadi. `[7, 14, 21, 28, 35, 42, 49, 56, 63, 70]`

Topik

- Langkah 1: Buat mesin negara
- Langkah 2: Buat fungsi Lambda
- Langkah 3: Jalankan mesin negara

Langkah 1: Buat mesin negara

Pada langkah ini, Anda membuat prototipe alur kerja mesin status yang meneruskan satu item dari sekumpulan item ke setiap pemanggilan fungsi Lambda yang akan Anda buat di Langkah 2.

- Gunakan definisi berikut untuk membuat mesin status menggunakan konsol Step Functions. Untuk informasi tentang membuat mesin status, lihat Langkah 1: Buat prototipe alur kerja di Memulai dengan menggunakan Distributed Map state tutorial.

  Di mesin status ini, Anda menentukan status Peta Terdistribusi yang menerima larik 10 bilangan bulat sebagai input dan meneruskan item array ini ke eksekusi alur kerja anak dalam batch. Setiap eksekusi alur kerja anak menerima batch tiga item sebagai input dan menjalankan status Peta Inline. Setiap iterasi status Peta Sebaris memanggil fungsi Lambda dan meneruskan item dari batch ke fungsi. Fungsi ini kemudian mengalikan item dengan faktor 7 dan mengembalikan hasilnya.

  Output dari setiap eksekusi alur kerja anak adalah array JSON yang berisi hasil perkalian untuk setiap item yang dilewatkan.

  > **Important**
  >
  > Pastikan untuk mengganti Amazon Resource Name (ARN) dari fungsi Lambda dalam kode berikut dengan ARN dari fungsi yang akan Anda buat di Langkah 2.

```
{
  "StartAt": "Pass",
  "States": {
    "Pass": {
      "Type": "Pass",
```

```
      "Next": "Map",
      "Result": {
        "MyMultiplicationFactor": 7,
        "MyItems": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
      }
    },
    "Map": {
      "Type": "Map",
      "ItemProcessor": {
        "ProcessorConfig": {
          "Mode": "DISTRIBUTED",
          "ExecutionType": "STANDARD"
        },
        "StartAt": "InnerMap",
        "States": {
          "InnerMap": {
            "Type": "Map",
            "ItemProcessor": {
              "ProcessorConfig": {
                "Mode": "INLINE"
              },
              "StartAt": "Lambda Invoke",
              "States": {
                "Lambda Invoke": {
                  "Type": "Task",
                  "Resource": "arn:aws:states:::lambda:invoke",
                  "OutputPath": "$.Payload",
                  "Parameters": {
                    "Payload.$": "$",
                    "FunctionName": "arn:aws:lambda:us-
east-1:123456789012:function:functionName"
                  },
                  "Retry": [
                    {
                      "ErrorEquals": [
                        "Lambda.ServiceException",
                        "Lambda.AWSLambdaException",
                        "Lambda.SdkClientException",
                        "Lambda.TooManyRequestsException"
                      ],
                      "IntervalSeconds": 2,
                      "MaxAttempts": 6,
                      "BackoffRate": 2
                    }
```

```
                ],
                "End": true
              }
            }
          },
          "End": true,
          "ItemsPath": "$.Items",
          "ItemSelector": {
            "MyMultiplicationFactor.$": "$.BatchInput.MyMultiplicationFactor",
            "MyItem.$": "$$.Map.Item.Value"
          }
        }
      }
    },
    "End": true,
    "Label": "Map",
    "MaxConcurrency": 1000,
    "ItemsPath": "$.MyItems",
    "ItemBatcher": {
      "MaxItemsPerBatch": 3,
      "BatchInput": {
        "MyMultiplicationFactor.$": "$.MyMultiplicationFactor"
      }
    }
  }
}
```

Langkah 2: Buat fungsi Lambda

Pada langkah ini, Anda membuat fungsi Lambda yang memproses setiap item yang diteruskan dari batch.

> **Important**
>
> Pastikan fungsi Lambda Anda Wilayah AWS sama dengan mesin status Anda.

Untuk membuat fungsi Lambda

1. Gunakan konsol Lambda untuk membuat nama fungsi Lambda Python 3.9. **`ProcessSingleItem`** Untuk informasi tentang membuat fungsi Lambda, lihat Langkah 4: Mengkonfigurasi fungsi Lambda dalam Memulai dengan menggunakan Distributed Map state tutorial.
2. Salin kode berikut untuk fungsi Lambda dan tempelkan ke bagian sumber Kode fungsi Lambda Anda.

```
import json

def lambda_handler(event, context):

    multiplication_factor = event['MyMultiplicationFactor']
    item = event['MyItem']

    result = multiplication_factor * item

    return {
        'statusCode': 200,
        'multiplied': result
    }
```

3. Setelah Anda membuat fungsi Lambda Anda, salin ARN fungsi yang ditampilkan di sudut kanan atas halaman. Untuk menyalin ARN, klik tombol.

Berikut ini adalah contoh ARN, di mana *`function-name`*adalah nama fungsi Lambda (dalam hal ini,): `ProcessSingleItem`

```
arn:aws:lambda:us-east-1:123456789012:function:function-name
```

Anda harus menyediakan fungsi ARN di mesin status yang Anda buat di Langkah 1.

4. Pilih Terapkan untuk menyebarkan perubahan.

Langkah 3: Jalankan mesin negara

Saat Anda menjalankan mesin status, status Peta Terdistribusi memulai empat eksekusi alur kerja anak, di mana setiap eksekusi memproses tiga item, sementara satu eksekusi memproses satu item.

Contoh berikut menunjukkan data yang diteruskan ke salah satu pemanggilan `ProcessSingleItem`fungsi di dalam eksekusi alur kerja anak.

```
{
  "MyMultiplicationFactor": 7,
  "MyItem": 1
}
```

Dengan masukan ini, contoh berikut menunjukkan output yang dikembalikan oleh fungsi Lambda.

```
{
  "statusCode": 200,
  "multiplied": 7
}
```

Contoh berikut menunjukkan output JSON array untuk salah satu eksekusi alur kerja anak.

```
[
  {
    "statusCode": 200,
    "multiplied": 7
  },
  {
    "statusCode": 200,
    "multiplied": 14
  },
  {
    "statusCode": 200,
    "multiplied": 21
  }
]
```

Mesin state mengembalikan output berikut yang berisi empat array untuk empat eksekusi alur kerja anak. Array ini berisi hasil perkalian dari masing-masing item input.

Akhirnya, output mesin state adalah array bernama `multiplied` yang menggabungkan semua hasil perkalian yang dikembalikan untuk empat eksekusi alur kerja anak.

```
[
  [
    {
      "statusCode": 200,
```

```
      "multiplied": 7
    },
    {
      "statusCode": 200,
      "multiplied": 14
    },
    {
      "statusCode": 200,
      "multiplied": 21
    }
  ],
  [
    {
      "statusCode": 200,
      "multiplied": 28
    },
    {
      "statusCode": 200,
      "multiplied": 35
    },
    {
      "statusCode": 200,
      "multiplied": 42
    }
  ],
  [
    {
      "statusCode": 200,
      "multiplied": 49
    },
    {
      "statusCode": 200,
      "multiplied": 56
    },
    {
      "statusCode": 200,
      "multiplied": 63
    }
  ],
  [
    {
      "statusCode": 200,
      "multiplied": 70
    }
```

```
  ]
]
```

Untuk menggabungkan semua hasil perkalian yang dikembalikan oleh eksekusi alur kerja anak ke dalam satu larik keluaran, Anda dapat menggunakan bidang tersebut. ResultSelector Tentukan bidang ini di dalam status Peta Terdistribusi untuk menemukan semua hasil, ekstrak hasil individual, dan kemudian menggabungkannya ke dalam satu array keluaran bernama`multiplied`.

Untuk menggunakan `ResultSelector` bidang, perbarui definisi mesin status Anda seperti yang ditunjukkan pada contoh berikut.

```
{
  "StartAt": "Pass",
  "States": {
    ...
    ...
    "Map": {
      "Type": "Map",
      ...
      ...
      "ItemBatcher": {
        "MaxItemsPerBatch": 3,
        "BatchInput": {
          "MyMultiplicationFactor.$": "$.MyMultiplicationFactor"
        }
      },
      "ItemsPath": "$.MyItems",
      "ResultSelector": {
        "multiplied.$": "$..multiplied"
      }
    }
  }
}
```

Mesin status diperbarui mengembalikan array output konsolidasi seperti yang ditunjukkan pada contoh berikut.

```
{
  "multiplied": [7, 14, 21, 28, 35, 42, 49, 56, 63, 70]
}
```

Memulai Eksekusi Mesin Status dalam Respons terhadap Peristiwa Amazon S3

Anda dapat menjalankan mesin AWS Step Functions status sebagai respons terhadap EventBridge aturan Amazon.

Tutorial ini menunjukkan cara mengkonfigurasi mesin status sebagai target untuk EventBridge aturan Amazon. Aturan ini akan memulai eksekusi mesin status saat file ditambahkan ke bucket Amazon Simple Storage Service (Amazon S3).

Untuk aplikasi yang praktis, Anda dapat meluncurkan mesin status yang melakukan operasi pada file yang Anda tambahkan ke bucket, seperti membuat thumbnail atau menjalankan analisis Amazon Rekognition pada file citra dan video.

Dalam tutorial ini, Anda memulai eksekusi mesin `Helloworld` status dengan mengunggah file ke bucket Amazon S3. Kemudian Anda meninjau contoh input eksekusi tersebut untuk mengidentifikasi informasi yang disertakan dalam input dari pemberitahuan peristiwa Amazon S3 yang dikirimkan ke. EventBridge

Topik

- Prasyarat: Buat Mesin Status
- Langkah 1: Buat Bucket di Amazon S3
- Langkah 2: Aktifkan Pemberitahuan Acara Amazon S3 dengan EventBridge
- Langkah 3: Buat EventBridge Aturan Amazon
- Langkah 4: Uji Aturan
- Contoh Input Eksekusi

Prasyarat: Buat Mesin Status

Sebelum Anda dapat mengonfigurasi mesin status sebagai EventBridge target Amazon, Anda harus membuat mesin status.

- Untuk membuat mesin state dasar, gunakan mesin Creating state yang menggunakan tutorial fungsi Lambda.
- Jika Anda sudah memiliki mesin `Helloworld` negara, lanjutkan ke langkah berikutnya.

Langkah 1: Buat Bucket di Amazon S3

Sekarang setelah Anda memiliki mesin `Helloworld` negara, Anda perlu membuat bucket Amazon S3 yang menyimpan file Anda. Pada Langkah 3 tutorial ini, Anda menyiapkan aturan sehingga ketika file diunggah ke bucket ini, EventBridge memicu eksekusi mesin status Anda.

1. Arahkan ke konsol Amazon S3, lalu pilih Buat bucket untuk membuat bucket tempat Anda ingin menyimpan file dan memicu aturan acara Amazon S3.
2. Masukkan Nama bucket, seperti *`username`*`-sfn-tutorial`.

   > **Note**
   >
   > Nama bucket harus unik di semua nama bucket yang ada di semua AWS Wilayah dalam Amazon S3. Gunakan *`nama pengguna`* Anda sendiri untuk membuat nama ini unik. Anda perlu membuat semua sumber daya di AWS Wilayah yang sama.

3. Simpan semua pilihan default di halaman, dan pilih Buat ember.

Langkah 2: Aktifkan Pemberitahuan Acara Amazon S3 dengan EventBridge

Setelah Anda membuat bucket Amazon S3, konfigurasikan bucket tersebut untuk mengirim peristiwa ke EventBridge setiap kali peristiwa tertentu terjadi di bucket S3 Anda, seperti unggahan file.

1. Arahkan ke konsol Amazon S3.
2. Di Bucket pilih nama bucket yang ingin Anda aktifkan untuk acara.
3. Pilih Properti.
4. Gulir ke bawah halaman untuk melihat bagian Pemberitahuan Acara, lalu pilih Edit di EventBridge subbagian Amazon.
5. Di bawah Kirim pemberitahuan ke Amazon EventBridge untuk semua acara di bucket ini, pilih Aktif.
6. Pilih Save changes (Simpan perubahan).

   > **Note**
   >
   > Setelah Anda mengaktifkan EventBridge, dibutuhkan sekitar lima menit agar perubahan diterapkan.

Langkah 3: Buat EventBridge Aturan Amazon

Setelah Anda memiliki mesin status, dan telah membuat bucket Amazon S3 dan mengonfigurasinya untuk mengirim pemberitahuan acara EventBridge, buat aturan. EventBridge

> **Note**
>
> Anda harus mengonfigurasi EventBridge aturan di AWS Wilayah yang sama dengan bucket Amazon S3.

Untuk membuat aturan

1. Arahkan ke EventBridge konsol Amazon, pilih Buat aturan.

   > **Tip**
   >
   > Atau, di panel navigasi di EventBridge konsol, pilih Aturan di bawah Bus, lalu pilih Buat aturan.

2. Masukkan Nama untuk aturan Anda (misalnya,*S3Step Functions*) dan secara opsional masukkan Deskripsi untuk aturan tersebut.
3. Untuk bus Acara dan tipe Aturan, pertahankan pilihan default.
4. Pilih Selanjutnya. Ini membuka halaman pola acara Build.
5. Gulir ke bawah ke bagian Pola acara, dan lakukan hal berikut:

   a. Untuk sumber Acara, pertahankan pilihan default AWSacara atau acara EventBridge mitra.

   b. Untuk AWSlayanan, pilih Simple Storage Service (S3).

   c. Untuk jenis Acara, pilih Pemberitahuan Acara Amazon S3.

   d. Pilih peristiwa tertentu, lalu pilih Object Created.

   e. Pilih Bucket spesifik berdasarkan nama dan masukkan nama bucket yang Anda buat di Langkah 1 (*username*`-sfn-tutorial`) untuk menyimpan file Anda.

   f. Pilih Selanjutnya. Ini membuka halaman Pilih target.

Untuk membuat target

1. Di Target 1, pertahankan pilihan AWSlayanan default.
2. Dalam daftar tarik-turun Pilih target, pilih mesin status Step Functions.
3. Dalam daftar mesin Negara, pilih mesin status yang Anda buat sebelumnya (misalnya,`Helloworld`).
4. Simpan semua pilihan default di halaman, dan pilih Berikutnya. Ini membuka halaman Konfigurasi tag.
5. Pilih Selanjutnya sekali lagi. Ini membuka halaman Review dan create.
6. Tinjau detail aturan dan pilih Buat aturan.

   Aturan dibuat dan halaman Aturan ditampilkan, mencantumkan semua EventBridge aturan Amazon Anda.

Langkah 4: Uji Aturan

Sekarang semua sudah berada di tempatnya, uji penambahan file ke bucket Amazon S3, lalu lihat input eksekusi mesin status yang dihasilkan.

1. Tambahkan file ke bucket Amazon S3.

   Arahkan ke konsol Amazon S3, pilih bucket yang Anda buat untuk menyimpan file (***username*-sfn-tutorial**), lalu pilih Unggah.
2. Tambahkan file, misalnya*`test.png`*, lalu pilih Unggah.

   Tindakan ini meluncurkan eksekusi mesin status Anda, meneruskan informasi dari AWS CloudTrail sebagai input.
3. Periksa eksekusi untuk mesin status Anda.

   Arahkan ke konsol Step Functions dan pilih mesin status yang digunakan dalam EventBridge aturan Amazon Anda (`Helloworld`).
4. Pilih eksekusi terbaru dari mesin status itu dan perluas bagian Input Eksekusi.

   Input ini mencakup informasi seperti nama bucket dan nama objek. Dalam kasus penggunaan dunia nyata, mesin status dapat menggunakan input ini untuk melakukan tindakan pada objek tersebut.

Contoh Input Eksekusi

Contoh berikut menunjukkan input khas untuk eksekusi mesin negara.

```
{
    "version": "0",
    "id": "6c540ad4-0671-9974-6511-756fbd7771c3",
    "detail-type": "Object Created",
    "source": "aws.s3",
    "account": "123456789012",
    "time": "2023-06-23T23:45:48Z",
    "region": "us-east-2",
    "resources": [
      "arn:aws:s3:::username-sfn-tutorial"
    ],
    "detail": {
      "version": "0",
      "bucket": {
        "name": "username-sfn-tutorial"
      },
      "object": {
        "key": "test.png",
        "size": 800704,
        "etag": "f31d8546bb67845b4d3048cde533b937",
        "sequencer": "00621049BA9A8C712B"
      },
      "request-id": "79104EXAMPLEB723",
      "requester": "123456789012",
      "source-ip-address": "200.0.100.11",
      "reason": "PutObject"
    }
  }
```

Membuat Step Functions API menggunakan API Gateway

Anda dapat menggunakan Amazon API Gateway untuk mengaitkan AWS Step Functions API Anda dengan metode di API Gateway API. Saat permintaan HTTPS dikirim ke metode API, API Gateway akan memanggil tindakan API Step Functions Anda.

Tutorial ini menunjukkan cara membuat API yang menggunakan satu sumber daya dan metode `POST` untuk berkomunikasi dengan tindakan API `StartExecution`. Anda akan menggunakan konsol

AWS Identity and Access Management (IAM) untuk membuat peran untuk API Gateway. Kemudian, Anda akan menggunakan konsol API Gateway untuk membuat API Gateway API, membuat sumber daya dan metode, serta memetakan metode ke tindakan API `StartExecution`. Akhirnya, Anda akan men-deploy dan menguji API Anda.

> **Note**
>
> Meskipun Amazon API Gateway dapat memulai eksekusi Step Functions dengan memanggil `StartExecution`, Anda harus memanggil `DescribeExecution` untuk mendapatkan hasilnya.

Topik

- Langkah 1: Buat IAM role untuk API Gateway
- Langkah 2: Buat API Gateway API Anda
- Langkah 3: Uji dan Deploy API Gateway API

Langkah 1: Buat IAM role untuk API Gateway

Sebelum Anda membuat API Gateway API, Anda harus memberikan izin API Gateway untuk memanggil tindakan API Step Functions.

Untuk menyiapkan izin API Gateway

1. Masuk ke konsol IAM dan pilih Peran, Buat peran.
2. Pada halaman Pilih entitas tepercaya, lakukan hal berikut:
   a. Untuk jenis entitas Tepercaya, pertahankan pilihan default Layanan AWS.
   b. Untuk kasus Penggunaan, pilih API Gateway dari daftar tarik-turun.
3. Pilih API Gateway, lalu pilih Berikutnya.
4. Pada halaman Tambahkan izin, pilih Berikutnya.
5. (Opsional) Pada halaman Nama, tinjau, dan buat, masukkan detail, seperti nama peran. Misalnya, masukkan **`APIGatewayToStepFunctions`**.
6. Pilih Buat peran.

   IAM role muncul dalam daftar peran.

7. Pilih nama peran Anda dan perhatikan ARN Peran, seperti yang ditunjukkan dalam contoh berikut.

```
arn:aws:iam::123456789012:role/APIGatewayToStepFunctions
```

Untuk melampirkan kebijakan ke IAM role

1. Pada halaman Peran, cari peran Anda (`APIGatewayToStepFunctions`), kemudian pilih peran tersebut.
2. Pada tab Izin, pilih Tambahkan izin, lalu pilih Lampirkan kebijakan.
3. Pada halaman Lampirkan Kebijakan, cari`AWSStepFunctionsFullAccess`, pilih kebijakan, lalu pilih Tambahkan izin.

Langkah 2: Buat API Gateway API Anda

Setelah Anda membuat IAM role, Anda dapat membuat API kustom di API Gateway Anda.

Untuk membuat API

1. Buka konsol Amazon API Gateway, lalu pilih Buat API.
2. Pada halaman Pilih jenis API, di panel REST API, pilih Build.
3. Pada halaman Create REST API, pilih New API, lalu masukkan ***`StartExecutionAPI`*** untuk nama API.
4. Pertahankan jenis endpoint API sebagai Regional, lalu pilih Create API.

Untuk membuat sumber daya

1. Pada halaman Sumber Daya StartExecution ***`API`***, pilih Buat sumber daya.
2. Pada halaman Buat sumber daya, masukkan **`execution`** nama Sumber Daya, lalu pilih Buat sumber daya.

Untuk membuat metode POST

1. Pilih sumber daya /eksekusi, lalu pilih Create method.
2. Untuk jenis Metode, pilihPOST.
3. Untuk jenis Integrasi, pilih AWS layanan.
4. Untuk Wilayah AWS, pilih Wilayah dari daftar.

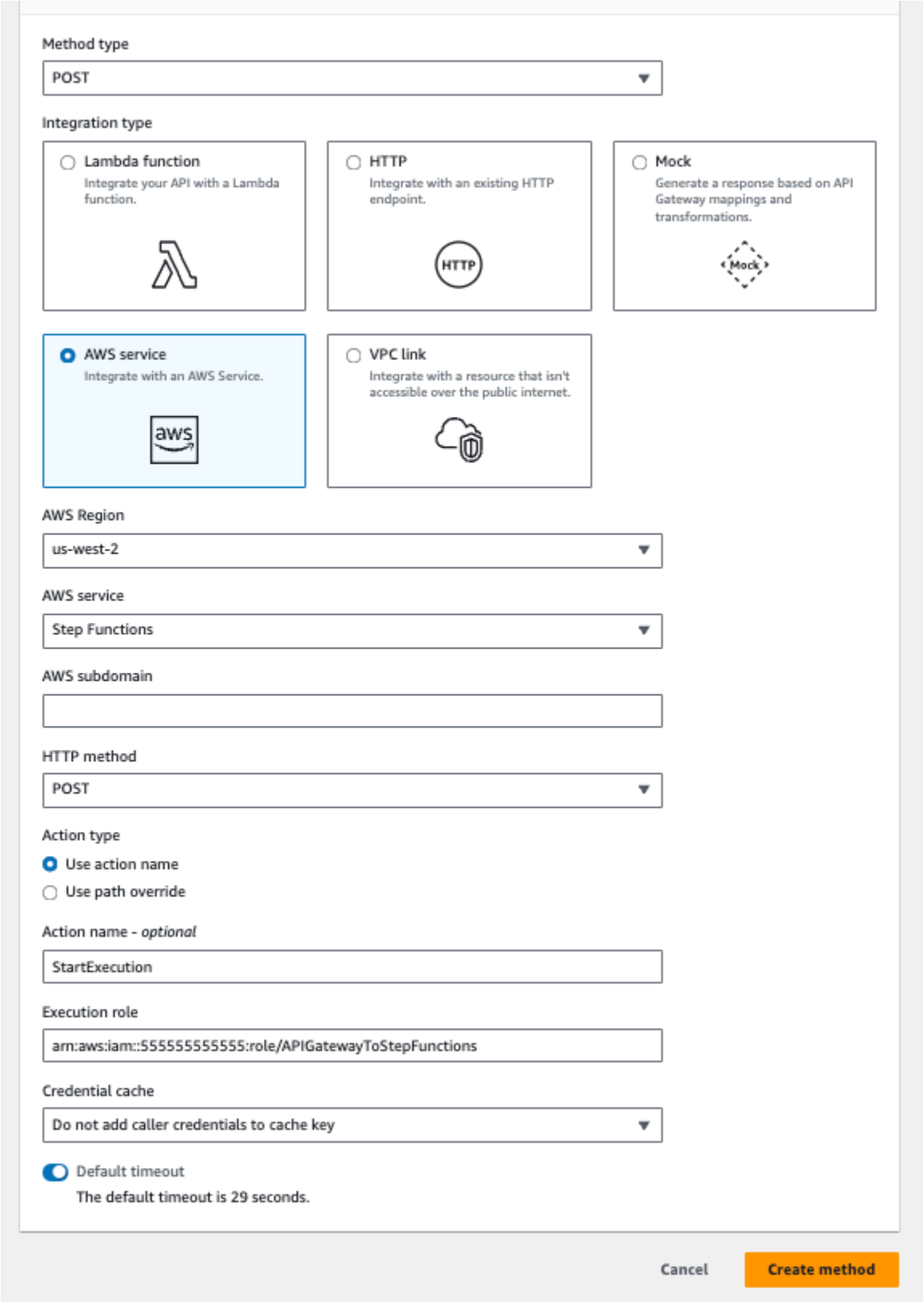
> **Note**
>
> Untuk Wilayah yang saat ini mendukung Step Functions, lihat Wilayah yang Didukung.

5. Untuk Layanan AWS, pilih Step Functions dari daftar.
6. Biarkan AWS subdomain kosong.
7. Untuk metode HTTP, pilih POST dari daftar.

> **Note**
>
> Semua tindakan API Step Functions menggunakan metode POST HTTP.

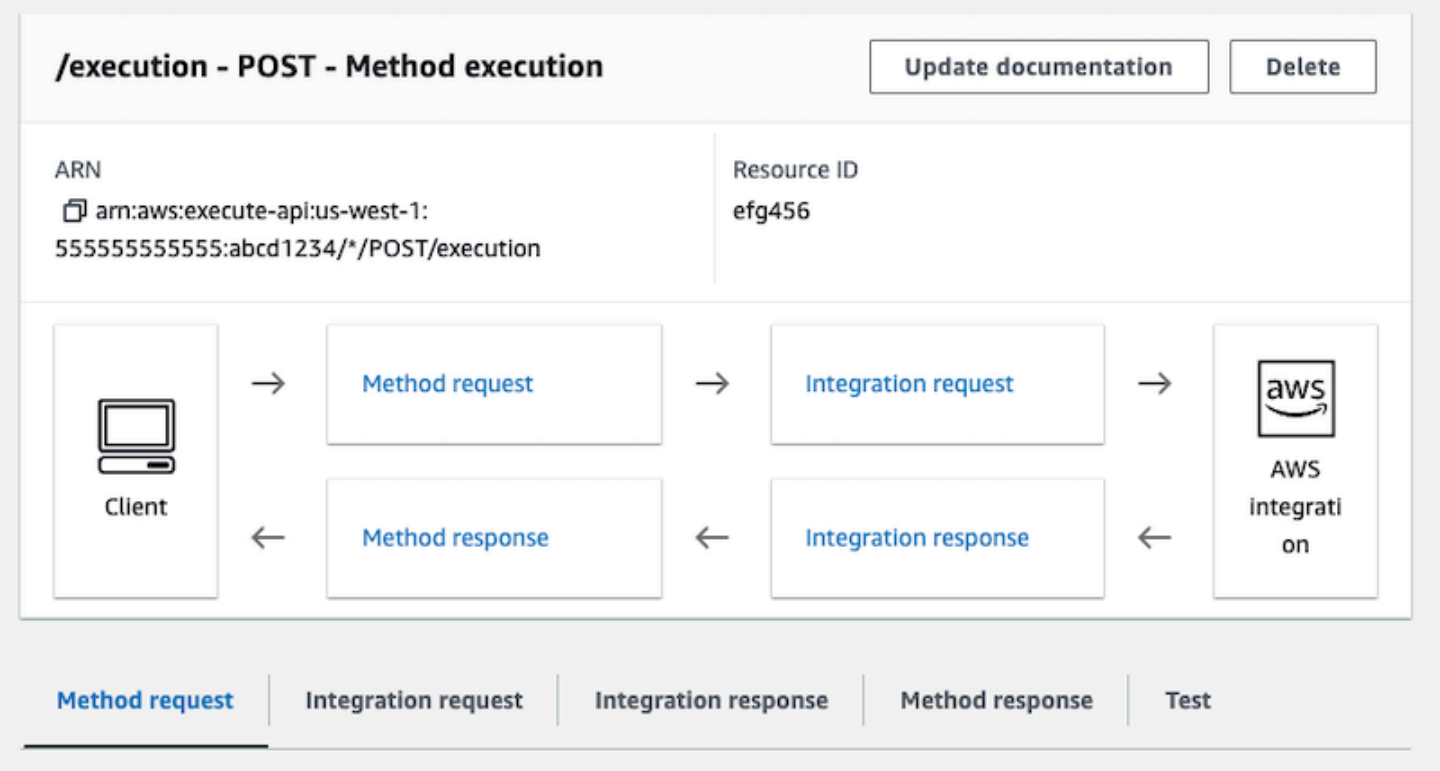
8. Untuk jenis tindakan, pilih Gunakan nama tindakan.
9. Untuk nama Action, masukkan**`StartExecution`**.
10. Untuk peran Eksekusi, masukkan peran ARN dari peran IAM yang Anda buat sebelumnya, seperti yang ditunjukkan pada contoh berikut.

```
arn:aws:iam::123456789012:role/APIGatewayToStepFunctions
```



11. Simpan opsi default untuk cache Credential dan batas waktu default, lalu pilih Simpan.

Pemetaan visual antara API Gateway dan Step Functions ditampilkan pada halaman eksekusi / execution - POST - Method.



Langkah 3: Uji dan Deploy API Gateway API

Setelah Anda selesai membuat API, uji dan deploy API.

Untuk menguji komunikasi antara API Gateway dan Step Functions

1. Pada halaman /execution - POST - Method Execution, pilih tab Test. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Pada tab /execution - POST - Method Test, salin parameter permintaan berikut ke bagian badan Permintaan menggunakan ARN dari mesin status yang ada (atau buat mesin status baru yang menggunakan fungsi Lambda), lalu pilih Uji.

```
{
   "input": "{}",
   "name": "MyExecution",
   "stateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:HelloWorld"
}
```

Untuk informasi selengkapnya, lihat Sintaksis Permintaan `StartExecution` dalam Referensi API AWS Step Functions .

> **Note**
>
> Jika Anda tidak ingin menyertakan ARN mesin status Anda di badan panggilan API Gateway Anda, Anda dapat mengonfigurasi templat pemetaan di tab Permintaan integrasi, seperti yang ditunjukkan pada contoh berikut.
>
> ```
> {
>     "input": "$util.escapeJavaScript($input.json('$'))",
>     "stateMachineArn": "$util.escapeJavaScript($stageVariables.arn)"
> }
> ```
>
> Dengan pendekatan ini, Anda dapat menentukan ARN dari mesin negara yang berbeda berdasarkan tahap pengembangan Anda (misalnya,, `devtest`, dan`prod`). Untuk informasi selengkapnya tentang menentukan variabel tahap dalam template pemetaan, lihat `$stageVariables`di Panduan Pengembang API Gateway.

3. Eksekusi dimulai dan ARN eksekusi dan tanggal zamannya ditampilkan di bawah badan Response.

```
{
   "executionArn": "arn:aws:states:us-
east-1:123456789012:execution:HelloWorld:MyExecution",
   "startDate": 1486768956.878
}
```

> **Note**
>
> Anda dapat melihat eksekusi dengan memilih mesin status Anda di konsol AWS Step Functions.

Untuk men-deploy API Anda

1. Pada halaman Sumber Daya StartExecution ***API***, pilih Deploy API.

2. Untuk Stage, pilih New stage.
3. Untuk nama Panggung, masukkan**`alpha`**.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Pilih Deploy.

Untuk menguji deployment Anda

1. Pada halaman Stages StartExecution ***API***, perluas alpha,/, /execution, POST, lalu pilih metode POST.
2. Di bawah Metode penggantian, pilih ikon salin untuk menyalin URL pemanggilan API Anda. URL lengkap akan terlihat seperti contoh berikut.

```
https://a1b2c3d4e5.execute-api.us-east-1.amazonaws.com/alpha/execution
```

3. Dari baris perintah, jalankan perintah `curl` menggunakan ARN dari mesin status Anda, kemudian minta URL dari deployment Anda, seperti yang ditunjukkan dalam contoh berikut.

```
curl -X POST -d '{"input": "{}","name": "MyExecution","stateMachineArn":
 "arn:aws:states:us-east-1:123456789012:stateMachine:HelloWorld"}' https://
a1b2c3d4e5.execute-api.us-east-1.amazonaws.com/alpha/execution
```

ARN eksekusi dan tanggal jangka waktunya dikembalikan, seperti berikut ini.

```
{"executionArn":"arn:aws:states:us-
east-1:123456789012:execution:HelloWorld:MyExecution","startDate":1.486772644911E9}
```

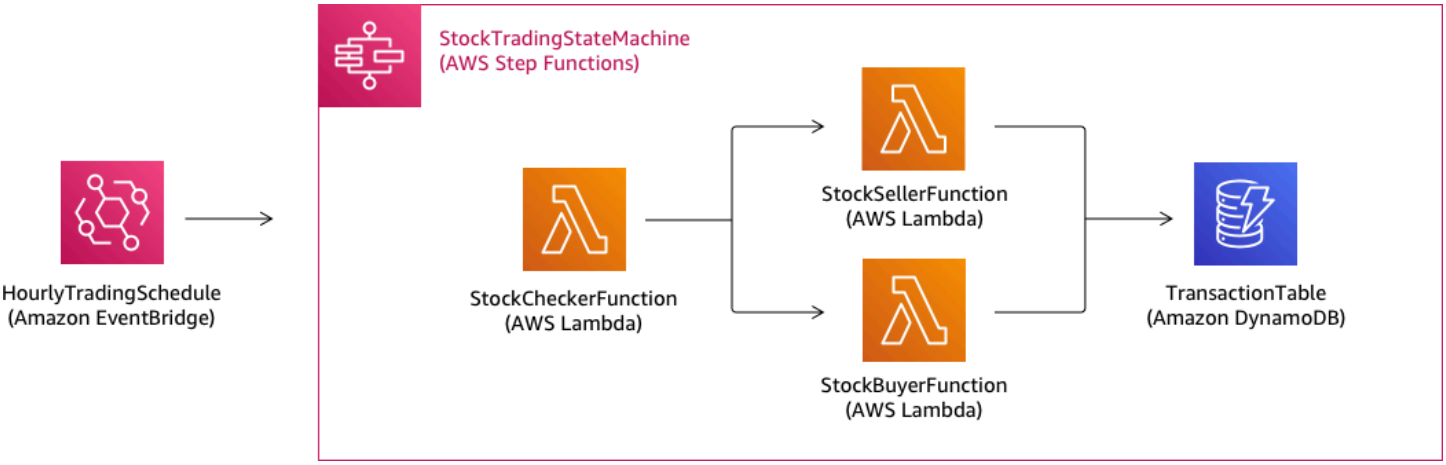
> **Note**
>
> Jika Anda mendapatkan kesalahan “Token Otentikasi Hilang”, pastikan URL pemanggilan diakhiri dengan /execution.

Membuat mesin kondisi Step FunctionsAWS SAM

Dalam panduan ini, Anda mengunduh, membangun, dan men-deploy contoh aplikasi AWS SAM yang berisi mesin status AWS Step Functions. Aplikasi ini membuat alur kerja perdagangan saham tiruan

yang berjalan pada jadwal yang telah ditentukan sebelumnya (perhatikan bahwa jadwal dinonaktifkan secara default untuk menghindari biaya).

Diagram berikut menunjukkan komponen dari aplikasi ini:



Berikut ini adalah pratinjau perintah yang Anda jalankan untuk membuat aplikasi sampel Anda. Untuk detail selengkapnya tentang masing-masing perintah ini, lihat bagian selanjutnya di halaman ini

```
# Step 1 - Download a sample application. For this tutorial you
#   will follow the prompts to select an AWS Quick Start Template
#   called 'Multi-step workflow'
sam init

# Step 2 - Build your application
cd project-directory
sam build

# Step 3 - Deploy your application
sam deploy --guided
```

Prasyarat

Panduan ini mengasumsikan bahwa Anda telah menyelesaikan langkah-langkah dalam Menginstal CLI AWS SAM untuk OS Anda. Panduan ini mengasumsikan bahwa Anda telah melakukan hal berikut:

1. Membuat akun AWS.
2. Mengonfigurasi izin IAM.
3. Menginstal Homebrew. Catatan: Homebrew hanyalah prasyarat untuk Linux dan macOS.

4. Menginstal CLI AWS SAM. Catatan: Pastikan bahwa Anda memiliki versi 0.52.0 atau yang lebih baru. Anda dapat memeriksa versi yang Anda miliki dengan mengeksekusi perintah `sam --version`.

Langkah 1: Unduh Sampel Aplikasi AWS SAM

Perintah untuk menjalankan:

```
sam init
```

Ikuti perintah di layar untuk memilih hal berikut:

1. Templat: Templat Quick Start AWS
2. Bahasa: Python, Ruby, NodeJS, Go, Java, atau .NET
3. Nama proyek: (nama pilihan Anda - yang default adalah `sam-app`)
4. Memulai aplikasi cepat:Alur kerja multi-step

ApaAWS SAMsedang melakukan:

Perintah ini membuat direktori dengan nama yang Anda berikan untuk prompt 'Project name' (default-nya adalah `sam-app`). Isi spesifik direktori akan tergantung pada bahasa yang Anda pilih.

Berikut adalah isi direktori saat Anda memilih salah satu dari waktu aktif Python:

```
### README.md
### functions
#   ### __init__.py
#   ### stock_buyer
#   #   ### __init__.py
#   #   ### app.py
#   #   ### requirements.txt
#   ### stock_checker
#   #   ### __init__.py
#   #   ### app.py
#   #   ### requirements.txt
#   ### stock_seller
#       ### __init__.py
#       ### app.py
#       ### requirements.txt
```

```
### statemachine
#   ### stock_trader.asl.json
### template.yaml
### tests
    ### unit
        ### __init__.py
        ### test_buyer.py
        ### test_checker.py
        ### test_seller.py
```

Ada dua file yang sangat menarik yang dapat Anda lihat:

- `template.yaml`: Berisi templat AWS SAM yang menentukan sumber daya aplikasi AWS Anda.
- `statemachine/stockTrader.asl.json`: Berisi ketentuan mesin status aplikasi, yang ditulis dalam Amazon States Language.

Anda dapat melihat entri berikut dalam file `template.yaml`, yang menunjuk ke file ketentuan mesin status:

```
    Properties:
      DefinitionUri: statemachine/stock_trader.asl.json
```

Akan sangat membantu untuk menjaga definisi mesin status sebagai file terpisah alih-alih menyematkannya diAWS SAMtemplat. Misalnya, melacak perubahan pada definisi mesin status lebih mudah jika Anda tidak menyertakan definisi dalam templat. Anda dapat menggunakan Workflow Studio untuk membuat dan memelihara definisi mesin status, dan mengekspor definisi dari konsol langsung ke file spesifikasi Amazon States Language tanpa menggabungkannya ke dalam template.

Untuk informasi selengkapnya tentang aplikasi sampel, lihat file `README.md` dalam direktori proyek.

Langkah 2: Bangun Aplikasi Anda

Perintah untuk menjalankan:

Pertama-tama ubah ke direktori proyek (yaitu, direktori tempat file `template.yaml` untuk aplikasi sampel berada; secara default adalah `sam-app`), lalu jalankan perintah ini:

```
sam build
```

Output contoh:

```
Build Succeeded

Built Artifacts  : .aws-sam/build
Built Template   : .aws-sam/build/template.yaml

Commands you can use next
=========================
[*] Invoke Function: sam local invoke
[*] Deploy: sam deploy --guided
```

ApaAWS SAMsedang melakukan:

CLI AWS SAM hadir dengan abstraksi untuk sejumlah waktu aktif Lambda guna membangun dependensi Anda, dan menyalin semua artefak build ke folder persiapan sehingga semuanya siap untuk dikemas dan di-deploy. Perintah `sam build` membangun setiap dependensi yang aplikasi Anda miliki, dan menyalin artefak build ke folder di bawah `.aws-sam/build`.

Langkah 3: Deploy Aplikasi Anda ke Cloud AWS

Perintah untuk menjalankan:

```
sam deploy --guided
```

Ikuti petunjuk di layar. Anda hanya dapat merespons dengan `Enter` untuk menerima opsi default yang disediakan dalam pengalaman interaktif.

ApaAWS SAMsedang melakukan:

Perintah ini men-deploy aplikasi Anda ke cloud AWS. Aplikasi tersebut mengambil artefak deployment yang Anda buat dengan perintah `sam build`, mengemas dan mengunggahnya ke bucket Amazon S3 yang dibuat oleh CLI AWS SAM, dan men-deploy aplikasi menggunakan AWS CloudFormation. Dalam output dari perintah deploy, Anda dapat melihat perubahan yang dilakukan pada tumpukan AWS CloudFormation Anda.

Anda dapat memverifikasi contoh mesin status Step Functions berhasil di-deploy dengan mengikuti langkah-langkah berikut:

1. Masuk ke AWS Management Console dan buka konsol Step Functions di https://console.aws.amazon.com/states/.
2. Di navigasi sebelah kiri, pilih Mesin status.
3. Temukan dan pilih mesin status baru Anda dalam daftar tersebut. Ini akan diberi nama StockTradingStateMachine-*<unique-hash>*.
4. Pilih tab Ketentuan.

Anda sekarang akan melihat representasi visual dari mesin status Anda. Anda dapat memverifikasi bahwa representasi visual sesuai dengan ketentuan mesin status yang ditemukan di dalam file `statemachine/stockTrader.asl.json` dari direktori proyek Anda.

Pemecahan Masalah

Kesalahan CLI SAM: "opsi tersebut tidak ada: --dipandu"

Saat mengeksekusi `sam deploy`, Anda akan melihat kesalahan berikut:

```
Error: no such option: --guided
```

Ini berarti Anda menggunakan versi lama dari CLI AWS SAM yang tidak mendukung parameter `--guided`. Untuk memperbaikinya, Anda dapat memperbarui versi CLI AWS SAM ke 0.33.0 atau yang lebih baru, atau menghilangkan parameter `--guided` dari perintah `sam deploy`.

Kesalahan CLI SAM: "Gagal untuk membuat sumber daya terkelola: Tidak dapat menemukan kredensial"

Saat mengeksekusi `sam deploy`, Anda akan melihat kesalahan berikut:

```
Error: Failed to create managed resources: Unable to locate credentials
```

Ini berarti Anda belum menyiapkan kredensial AWS untuk mengaktifkan CLI AWS SAM untuk melakukan panggilan layanan AWS. Untuk memperbaikinya, Anda harus menyiapkan kredensial

AWS. Untuk informasi selengkapnya, lihat Menyiapkan Kredensial AWS dalam Panduan Developer AWS Serverless Application Model.

Bersihkan

Jika Anda tidak lagi membutuhkan sumber daya AWS yang Anda buat dengan menjalankan tutorial ini, Anda dapat menghapusnya dengan menghapus tumpukan AWS CloudFormation yang Anda deploy.

Untuk menghapus tumpukan AWS CloudFormation yang dibuat dengan tutorial ini menggunakan AWS Management Console, ikuti langkah-langkah ini:

1. Masuk ke AWS Management Console dan buka konsol AWS CloudFormation di https://console.aws.amazon.com/cloudformation.
2. Di panel navigasi sebelah kiri, pilih Tumpukan.
3. Dalam daftar tumpukan, pilih aplikasi sam (atau nama tumpukan yang Anda buat).
4. Pilih Delete (Hapus).

Setelah selesai, status dari tumpukan akan berubah menjadi DELETE_COMPLETE.

Sebagai alternatif, Anda dapat menghapus tumpukan AWS CloudFormation dengan mengeksekusi perintah AWS CLI berikut:

```
aws cloudformation delete-stack --stack-name sam-app --region region
```

Verifikasi Tumpukan yang Dihapus

Untuk kedua metode menghapus tumpukan AWS CloudFormation, Anda dapat memverifikasi bahwa tumpukan tersebut telah dihapus dengan membuka https://console.aws.amazon.com/cloudformation, memilih Tumpukan di panel navigasi sebelah kiri, dan memilih Dihapus di menu tarik-turun di sebelah kanan kotak teks pencarian. Anda akan melihat nama tumpukan Anda aplikasi sam-app (atau nama tumpukan yang Anda buat) dalam daftar tumpukan yang dihapus.

Membuat mesin status Aktivitas menggunakan Step Functions

Tutorial ini menunjukkan cara membuat mesin status berbasis aktivitas menggunakan Java dan AWS Step Functions. Aktivitas mengizinkan Anda untuk mengontrol kode pekerja yang berjalan di tempat lain dari mesin status Anda. Untuk gambaran umum, lihat Aktivitas di Cara Step Functions bekerja.

Untuk menyelesaikan tutorial ini, Anda memerlukan hal berikut:

- SDK for Java. Contoh aktivitas dalam tutorial ini adalah aplikasi Java yang menggunakan AWS SDK for Java untuk berkomunikasi dengan AWS.
- AWS kredensil di lingkungan atau dalam file AWS konfigurasi standar. Untuk informasi selengkapnya, lihat Mengatur AWS Kredensil Anda di Panduan AWS SDK for Java Pengembang.

Topik

- Langkah 1: Buat Aktivitas
- Langkah 2: Buat mesin negara
- Langkah 3: Terapkan Pekerja
- Langkah 4: Jalankan mesin negara
- Langkah 5: Jalankan dan Hentikan Pekerja

Langkah 1: Buat Aktivitas

Anda harus membuat Step Functions mengetahui aktivitas yang pekerjanya (sebuah program) yang ingin Anda buat. Step Functions merespons dengan Amazon Resource Name (ARN) yang menetapkan identitas untuk aktivitas tersebut. Gunakan identitas ini untuk mengoordinasikan informasi yang diteruskan antara mesin status dan pekerja Anda.

> **Important**
>
> Pastikan tugas aktivitas Anda berada di bawah AWS akun yang sama dengan mesin status Anda.

1. Di konsol Step Functions, di panel navigasi yang ada di sebelah kiri, pilih Aktivitas.
2. Pilih Buat aktivasi.
3. Masukkan Nama untuk aktivitas, misalnya*get-greeting*, lalu pilih Buat aktivitas.
4. Ketika tugas aktivitas Anda dibuat, buat catatan ARN, seperti yang ditunjukkan dalam contoh berikut.

```
arn:aws:states:us-east-1:123456789012:activity:get-greeting
```

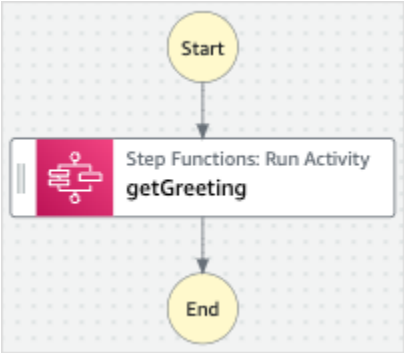
Langkah 2: Buat mesin negara

Buat mesin status yang menentukan waktu aktivitas Anda dipanggil dan waktu pekerja Anda harus melakukan pekerjaan utamanya, mengumpulkan hasilnya, dan mengembalikannya. Untuk membuat mesin status, Anda akan menggunakan Editor kode Workflow Studio.

1. Di dalam konsol Step Functions, di panel navigasi yang ada di sebelah kiri, pilih Mesin status.
2. Pada halaman State Machines, pilih Create state machine.
3. Dalam kotak dialog Pilih templat, pilih Kosong.
4. Pilih Pilih. Ini membuka Workflow Studio diMode desain.
5. Untuk tutorial ini, Anda akan menulis definisi Amazon States Language (ASL) dari mesin status Anda di editor kode. Untuk melakukan ini, pilih Kode.
6. Hapus kode boilerplate yang ada dan tempel kode berikut. Ingatlah untuk mengganti contoh ARN dalam kode ini dengan ARN dari tugas aktivitas yang Anda buat sebelumnya di bidang. `Resource`

```
{
  "Comment": "An example using a Task state.",
  "StartAt": "getGreeting",
  "Version": "1.0",
  "TimeoutSeconds": 300,
  "States":
  {
    "getGreeting": {
      "Type": "Task",
      "Resource": "arn:aws:states:us-east-1:123456789012:activity:get-greeting",
      "End": true
    }
  }
}
```

   Ini adalah deskripsi mesin negara Anda menggunakan Amazon States Language (ASL). ARN menentukan satu status `Task` bernama `getGreeting`. Untuk informasi selengkapnya, lihat Struktur Mesin Status.
7. PadaPanel visualisasi grafik, pastikan grafik alur kerja untuk definisi ASL yang Anda tambahkan terlihat mirip dengan grafik berikut.



8. Tentukan nama untuk mesin negara Anda. Untuk melakukan ini, pilih ikon edit di sebelah nama mesin status default MyStateMachine. Kemudian, dalam konfigurasi mesin Negara, tentukan nama di kotak Nama mesin Negara.

   Untuk tutorial ini, masukkan nama **`ActivityStateMachine`**.

9. (Opsional) Dalam konfigurasi mesin State, tentukan pengaturan alur kerja lainnya, seperti jenis mesin status dan peran pelaksanaannya.

   Untuk tutorial ini, simpan semua pilihan default di pengaturan mesin State.

   Jika sebelumnya Anda telah membuat peran IAM dengan izin yang benar untuk mesin status dan ingin menggunakannya, di Izin, pilih Pilih peran yang ada, lalu pilih peran dari daftar. Atau pilih Masukkan peran ARN dan kemudian berikan ARN untuk peran IAM itu.

10. Dalam kotak dialog Konfirmasi pembuatan peran, pilih Konfirmasi untuk melanjutkan.

    Anda juga dapat memilih Lihat pengaturan peran untuk kembali ke konfigurasi mesin Status.

> **Note**
>
> Jika Anda menghapus IAM role yang Step Functions buat, Step Functions tidak dapat membuatnya kembali nanti. Demikian pula, jika Anda mengubah peran (misalnya, dengan menghapus Step Functions dari principal dalam kebijakan IAM), Step Functions tidak dapat memulihkan pengaturan aslinya nanti.

Langkah 3: Terapkan Pekerja

Buat pekerja. Seorang pekerja adalah program yang bertanggung jawab untuk:

- Polling Step Functions untuk aktivitas menggunakan tindakan API `GetActivityTask`.

- Melakukan pekerjaan aktivitas menggunakan kode Anda, (misalnya, metode `getGreeting()` dalam kode berikut).
- Mengembalikan hasil menggunakan `SendTaskSuccess`, `SendTaskFailure`, dan tindakan API `SendTaskHeartbeat`.

> **Note**
>
> Untuk contoh pekerja aktivitas yang lebih lengkap, lihat Contoh Activity Worker di Ruby. Contoh ini memberikan implementasi berdasarkan praktik terbaik, yang dapat Anda gunakan sebagai referensi untuk pekerja aktivitas Anda. Kode menerapkan pola konsumen-produsen dengan jumlah utas yang dapat dikonfigurasi untuk poller dan pekerja aktivitas.

Untuk mengimplementasikan pekerja

1. Buat file bernama `GreeterActivities.java`.
2. Tambahkan kode berikut ke file tersebut.

```
import com.amazonaws.ClientConfiguration;
import com.amazonaws.auth.EnvironmentVariableCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.stepfunctions.AWSStepFunctions;
import com.amazonaws.services.stepfunctions.AWSStepFunctionsClientBuilder;
import com.amazonaws.services.stepfunctions.model.GetActivityTaskRequest;
import com.amazonaws.services.stepfunctions.model.GetActivityTaskResult;
import com.amazonaws.services.stepfunctions.model.SendTaskFailureRequest;
import com.amazonaws.services.stepfunctions.model.SendTaskSuccessRequest;
import com.amazonaws.util.json.Jackson;
import com.fasterxml.jackson.databind.JsonNode;
import java.util.concurrent.TimeUnit;


public class GreeterActivities {

    public String getGreeting(String who) throws Exception {
        return "{\"Hello\": \"" + who + "\"}";
    }

    public static void main(final String[] args) throws Exception {
        GreeterActivities greeterActivities = new GreeterActivities();
```

```
        ClientConfiguration clientConfiguration = new ClientConfiguration();
        clientConfiguration.setSocketTimeout((int)TimeUnit.SECONDS.toMillis(70));

        AWSStepFunctions client = AWSStepFunctionsClientBuilder.standard()
                .withRegion(Regions.US_EAST_1)
                .withCredentials(new EnvironmentVariableCredentialsProvider())
                .withClientConfiguration(clientConfiguration)
                .build();

        while (true) {
            GetActivityTaskResult getActivityTaskResult =
                    client.getActivityTask(
                            new
 GetActivityTaskRequest().withActivityArn(ACTIVITY_ARN));

            if (getActivityTaskResult.getTaskToken() != null) {
                try {
                    JsonNode json =
 Jackson.jsonNodeOf(getActivityTaskResult.getInput());
                    String greetingResult =

 greeterActivities.getGreeting(json.get("who").textValue());
                    client.sendTaskSuccess(
                            new SendTaskSuccessRequest().withOutput(

 greetingResult).withTaskToken(getActivityTaskResult.getTaskToken()));
                } catch (Exception e) {
                    client.sendTaskFailure(new
 SendTaskFailureRequest().withTaskToken(
                            getActivityTaskResult.getTaskToken()));
                }
            } else {
                Thread.sleep(1000);
            }
        }
    }
}
```

> **Note**
>
> Kelas `EnvironmentVariableCredentialsProvider` dalam contoh ini mengasumsikan bahwa variabel lingkungan `AWS_ACCESS_KEY_ID` (atau `AWS_ACCESS_KEY`) dan `AWS_SECRET_KEY` (atau `AWS_SECRET_ACCESS_KEY`)

ditetapkan. Untuk informasi selengkapnya tentang memberikan kredensil yang diperlukan ke pabrik, lihat AWSCredentialsProviderdi Referensi AWS SDK for Java API dan Menyiapkan AWS Kredensial dan Wilayah untuk Pengembangan di Panduan Pengembang.AWS SDK for Java
Secara default AWS SDK akan menunggu hingga 50 detik untuk menerima data dari server untuk operasi apa pun. Operasi `GetActivityTask` adalah operasi poll panjang yang akan menunggu hingga 60 detik untuk tugas yang tersedia berikutnya. Untuk mencegah menerima `SocketTimeoutException` kesalahan, atur SocketTimeout ke 70 detik.

3. Dalam daftar parameter konstruktor `GetActivityTaskRequest().withActivityArn()`, ganti nilai `ACTIVITY_ARN` dengan ARN dari tugas aktivitas yang Anda buat sebelumnya.

Langkah 4: Jalankan mesin negara

Saat Anda memulai eksekusi mesin status, pekerja Anda melakukan polling Step Functions untuk aktivitas, melakukan pekerjaannya (menggunakan input yang Anda berikan), dan mengembalikan hasilnya.

1. Pada ***`ActivityStateMachine`***halaman, pilih Mulai eksekusi.

   Kotak dialog Mulai eksekusi ditampilkan.
2. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

   a. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

   > **Note**
   >
   > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

   b. Dalam kotak Input, masukkan input JSON berikut untuk menjalankan alur kerja Anda.

   ```
   {
   ```

```
  "who": "AWS Step Functions"
}
```

c. Pilih Mulai Eksekusi.

d. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

   Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Langkah 5: Jalankan dan Hentikan Pekerja

Agar pekerja melakukan polling pada mesin status Anda untuk aktivitas, Anda harus menjalankan pekerja.

1. Pada baris perintah, navigasikan ke direktori tempat Anda membuat`GreeterActivities.java`.
2. Untuk menggunakan AWS SDK, tambahkan path lengkap `third-party` direktori `lib` dan ke dependensi file build Anda dan ke Java Anda. `CLASSPATH` Untuk informasi selengkapnya, lihat Mengunduh dan mengekstrak SDK dalam Panduan Developer AWS SDK for Java .
3. Kompilasikan file.

```
$ javac GreeterActivities.java
```

4. Jalankan file.

```
$ java GreeterActivities
```

5. Pada konsol Step Functions, navigasikan ke halaman Execution Details.
6. Ketika eksekusi selesai, periksa hasil eksekusi Anda.
7. Hentikan pekerja.

Ulangi loop dengan Lambda

Dalam tutorial ini, Anda menerapkan pola desain yang menggunakan mesin status dan fungsi AWS Lambda untuk mengiterasi putaran beberapa kali.

Gunakan pola desain ini setiap kali Anda perlu melacak jumlah putaran dalam mesin status. Implementasi ini dapat membantu Anda memecah tugas-tugas besar atau eksekusi berjalan lama menjadi potongan yang lebih kecil, atau untuk mengakhiri eksekusi setelah sejumlah peristiwa tertentu. Anda dapat menggunakan implementasi serupa untuk secara berkala mengakhiri dan memulai ulang eksekusi yang berjalan lama untuk menghindari melebihi kuota layanan untuk AWS Step Functions, AWS Lambda, atau layanan lainnya. AWS

Sebelum Anda mulai, ikuti Membuat mesin status Step Functions yang menggunakan Lambda tutorial untuk memastikan Anda terbiasa menggunakan Lambda dan Step Functions bersama-sama.

Langkah 1: Buat fungsi Lambda untuk mengulangi hitungan

Dengan menggunakan fungsi Lambda Anda dapat melacak jumlah iterasi dari putaran di mesin status Anda. Fungsi Lambda berikut menerima nilai input untuk `count`, `index`, dan `step`. Ia mengembalikan nilai-nilai ini dengan `index` diperbarui dan Boolean bernama `continue`. Fungsi Lambda mengatur `continue` ke `true` jika `index` kurang dari `count`.

Mesin status Anda kemudian mengimplementasikan status `Choice` yang mengeksekusi beberapa logika aplikasi jika `continue` adalah `true`, atau keluar jika statusnya adalah `false`.

Untuk membuat fungsi Lambda

1. Masuk ke konsol Lambda, lalu pilih Buat fungsi.
2. Pilih halaman Buat fungsi, pilih Penulis dari scratch.
3. Di bagian Informasi dasar, konfigurasikan fungsi Lambda Anda, sebagai berikut:
   a. Untuk Nama fungsi, masukkan `Iterator`.
   b. Untuk Runtime, pilih Node.js.
   c. Di Ubah peran eksekusi default, pilih Buat peran baru dengan izin Lambda dasar.
   d. Pilih Buat fungsi.
4. Salin kode berikut untuk fungsi Lambda ke sumber Kode.

```
export const handler = function (event, context, callback) {
```

```
  let index = event.iterator.index
  let step = event.iterator.step
  let count = event.iterator.count

  index = index + step

  callback(null, {
    index,
    step,
    count,
    continue: index < count
  })
}
```

Kode ini menerima nilai input untuk `count`, `index`, dan `step`. Kode ini menambahkan `index` dengan nilai dari `step` dan mengembalikan nilai-nilai ini, dan nilai Boolean `continue`. Nilai dari `continue` adalah `true` jika `index` kurang dari `count`.

5. Pilih Deploy.

Langkah 2: Uji Fungsi Lambda

Jalankan fungsi Lambda Anda dengan nilai-nilai numerik untuk melihatnya dalam operasi. Anda dapat memberikan nilai masukan untuk fungsi Lambda Anda yang meniru iterasi.

Untuk menguji fungsi Lambda Anda

1. Pilih Uji.
2. Dalam kotak dialog Konfigurasikan peristiwa uji, masukkan `TestIterator` di Nama acara kotak.
3. Ganti contoh data dengan hal berikut.

```
{
  "Comment": "Test my Iterator function",
  "iterator": {
    "count": 10,
    "index": 5,
    "step": 1
  }
}
```

Nilai-nilai ini meniru yang akan datang dari mesin status Anda selama iterasi. Fungsi Lambda akan meningkatkan indeks dan kembali `true` `continue` ketika indeks kurang dari. `count` Untuk uji ini, indeks telah bertambah menjadi 5. Tes akan meningkat `index` ke 6 dan diatur `continue` ke`true`.

4. Pilih Buat.
5. Pilih Uji untuk menguji fungsi Lambda Anda.

   Hasil tes ditampilkan di tab Hasil eksekusi.
6. Pilih tab Hasil eksekusi untuk melihat output.

```
{
  "index": 6,
  "step": 1,
  "count": 10,
  "continue": true
}
```

> **Note**
>
> Jika Anda mengatur `index` ke 9 dan menguji lagi, `index` kenaikan ke`10`, dan `continue` akan menjadi`false`.

Langkah 3: Buat Mesin Status

> **Sebelum Anda meninggalkan konsol Lambda...**
>
> Salin fungsi Lambda ARN. Tempelkan ke dalam catatan. Anda akan membutuhkannya di langkah berikutnya.

Selanjutnya, Anda akan membuat mesin negara dengan status berikut:

- `ConfigureCount`— Menetapkan nilai default untuk`count`,`index`, dan`step`.
- `Iterator`— Mengacu pada fungsi Lambda yang Anda buat sebelumnya, meneruskan nilai yang dikonfigurasi. `ConfigureCount`

- `IsCountReached`— Status pilihan yang melanjutkan loop atau melanjutkan ke `Done` status, berdasarkan nilai yang dikembalikan dari `Iterator` fungsi Anda.
- `ExampleWork`Sebuah rintisan untuk pekerjaan yang perlu dilakukan. Dalam contoh ini, alur kerja memiliki `Pass` status, tetapi dalam solusi nyata, Anda mungkin akan menggunakan. `Task`
- `Done`- Akhiri status alur kerja Anda.

Untuk membuat mesin status di konsol:

1. Buka Konsol Step Functions, lalu pilih Buat mesin status.

   > **Important**
   >
   > Mesin status Anda harus berada di AWS akun dan Wilayah yang sama dengan fungsi Lambda Anda.

2. Pilih template Kosong.
3. Di panel Kode, tempel JSON berikut yang mendefinisikan mesin status.

   Untuk informasi selengkapnya tentang Bahasa Status Amazon, lihat Struktur Mesin Status.

```
{
    "Comment": "Iterator State Machine Example",
    "StartAt": "ConfigureCount",
    "States": {

        "ConfigureCount": {
            "Type": "Pass",
            "Result": {
                "count": 10,
                "index": 0,
                "step": 1
            },
            "ResultPath": "$.iterator",
            "Next": "Iterator"
        },
        "Iterator": {
            "Type": "Task",
            "Resource": "arn:aws:lambda:us-east-1:123456789012:function:Iterate",
            "ResultPath": "$.iterator",
```

```
            "Next": "IsCountReached"
        },
        "IsCountReached": {
            "Type": "Choice",
            "Choices": [
                {
                    "Variable": "$.iterator.continue",
                    "BooleanEquals": true,
                    "Next": "ExampleWork"
                }
            ],
            "Default": "Done"
        },
        "ExampleWork": {
            "Comment": "Your application logic, to run a specific number of times",
            "Type": "Pass",
            "Result": {
              "success": true
            },
            "ResultPath": "$.result",
            "Next": "Iterator"
        },
        "Done": {
            "Type": "Pass",
            "End": true

        }
    }
}
```

4. Ganti `Iterator Resource` bidang dengan ARN untuk fungsi `Iterator` Lambda Anda yang Anda buat sebelumnya.
5. Pilih Config, dan masukkan Nama untuk mesin status Anda, seperti. *`IterateCount`*

> **Note**
>
> Nama mesin negara, eksekusi, dan tugas aktivitas tidak boleh melebihi 80 karakter panjangnya. Nama-nama ini harus unik untuk akun dan AWS Wilayah Anda, dan tidak boleh mengandung salah satu dari yang berikut:
>
> - Spasi putih
> - Karakter wildcard (? *)

- Karakter tanda kurung (< > { } [ ])
- Karakter khusus (" # % \ ^ | ~ ` $ & , ; : /)
- Karakter kontrol (\\u0000 - \\u001f atau \\u007f - \\u009f).

Jika mesin status Anda bertipe Express, Anda dapat memberikan nama yang sama untuk beberapa eksekusi mesin status. Step Functions menghasilkan ARN eksekusi unik untuk setiap eksekusi mesin status Express, bahkan jika beberapa eksekusi memiliki nama yang sama.
Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

6. Untuk Type, terima nilai default Standard. Untuk Izin, pilih Buat peran baru.
7. Pilih Buat, lalu Konfirmasikan kreasi peran.

Langkah 4: Mulai Eksekusi Baru

Setelah Anda membuat mesin status, Anda dapat memulai eksekusi.

1. Pada IterateCounthalaman, pilih Mulai eksekusi.
2. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

> **Note**
>
> Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

3. Pilih Mulai Eksekusi.

   Eksekusi baru mesin status Anda dimulai, dan halaman baru yang menunjukkan eksekusi Anda yang sedang berjalan akan ditampilkan.

Status tampilan grafik mesin, menunjukkan status Iterator berwarna biru untuk menunjukkan status kemajuan.

Eksekusi bertambah dalam beberapa langkah, melacak hitungan menggunakan fungsi Lambda Anda. Pada setiap iterasi, eksekusi melakukan pekerjaan contoh yang direferensikan dalam status `ExampleWork` di mesin status Anda.

Ketika hitungan mencapai jumlah yang ditentukan dalam status `ConfigureCount` di mesin status Anda, eksekusi berhenti dari iterasi dan berakhir.
Tampilan grafik mesin status, menunjukkan status Iterator dan status Selesai berwarna hijau untuk menunjukkan keduanya telah berhasil.

Melanjutkan Eksekusi Alur Kerja yang Berjalan Lama sebagai Eksekusi Baru

AWS Step Functions dirancang untuk menjalankan alur kerja yang memiliki durasi terbatas dan jumlah langkah. Eksekusi memiliki durasi maksimum satu tahun, dan maksimal 25.000 peristiwa (lihat Kuota).

Untuk eksekusi yang berjalan lama, agar tidak mencapai kuota keras 25.000 entri dalam riwayat peristiwa eksekusi, sebaiknya Anda memulai eksekusi alur kerja baru langsung dari status mesin status. `Task` Hal ini memungkinkan Anda untuk memecah alur kerja Anda menjadi mesin negara yang lebih kecil, dan untuk melanjutkan pekerjaan Anda yang sedang berlangsung dalam eksekusi baru. Untuk memulai eksekusi alur kerja ini, panggil tindakan `StartExecution` API dari `Task` status Anda dan teruskan parameter yang diperlukan.

Atau, Anda juga dapat menerapkan pola yang menggunakan fungsi Lambda untuk memulai eksekusi baru mesin status Anda untuk membagi pekerjaan yang sedang berlangsung di beberapa eksekusi alur kerja.

Tutorial ini menunjukkan kepada Anda kedua pendekatan untuk melanjutkan eksekusi alur kerja tanpa melebihi kuota layanan.

Topik

- Menggunakan aksi Step Functions API untuk melanjutkan eksekusi baru (disarankan)
- Menggunakan fungsi Lambda untuk melanjutkan eksekusi baru

Menggunakan aksi Step Functions API untuk melanjutkan eksekusi baru (disarankan)

Step Functions dapat memulai eksekusi alur kerja dengan memanggil API sendiri sebagai layanan terintegrasi. Kami menyarankan Anda menggunakan pendekatan ini untuk menghindari melebihi kuota layanan untuk eksekusi jangka panjang.

Langkah 1: Buat mesin status yang sudah berjalan lama

Buat mesin status yang sudah berjalan lama yang ingin Anda mulai dari `Task` keadaan mesin negara yang berbeda. Untuk tutorial ini, gunakan mesin state yang menggunakan fungsi Lambda.

> **Note**
>
> Pastikan untuk menyalin nama dan Nama Sumber Daya Amazon dari mesin status ini dalam file teks untuk digunakan nanti.

Langkah 2: Buat mesin status untuk memanggil aksi Step Functions API

Untuk memulai eksekusi alur kerja dari suatu negara **`Task`**

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Dalam kotak dialog Pilih templat, pilih Kosong.
3. Pilih Pilih. Ini membuka Workflow Studio diMode desain.
4. Dari tab Tindakan, seret aksi StartExecutionAPI dan letakkan pada status kosong berlabel Seret status pertama di sini.
5. Pilih StartExecutionstatus dan lakukan hal berikut di tab Konfigurasi diMode desain:

    a. Ubah nama negara menjadi**`Start nested execution`**.

    b. Untuk jenis Integrasi, pilih AWS SDK - baru dari daftar dropdown.

    c. Di Parameter API, lakukan hal berikut:

        i. Untuk`StateMachineArn`, ganti contoh Nama Sumber Daya Amazon dengan ARN mesin status Anda. Misalnya, masukkan ARN dari mesin negara yang menggunakan Lambda.

        ii. Untuk `Input` node, ganti teks placeholder yang ada dengan nilai berikut:

```
"Comment": "Starting workflow execution using a Step Functions API action"
```

iii. Pastikan input Anda di Parameter API terlihat mirip dengan yang berikut ini:

```
{
  "StateMachineArn": "arn:aws:states:us-
east-2:123456789012:stateMachine:LambdaStateMachine",
  "Input": {
    "Comment": "Starting workflow execution using a Step Functions API
 action",
    "AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID.$": "$$.Execution.Id"
  }
```

6. (Opsional) Pilih Definisi pada Inspector panel untuk melihat definisi alur kerja Anda yang dihasilkan secara otomatis Amazon States Language (ASL).

> **Tip**
>
> Anda juga dapat melihat definisi ASL di Editor kode Workflow Studio. Di editor kode, Anda juga dapat mengedit definisi ASL dari alur kerja Anda.

7. Tentukan nama untuk mesin negara Anda. Untuk melakukan ini, pilih ikon edit di sebelah nama mesin status default MyStateMachine. Kemudian, dalam konfigurasi mesin Negara, tentukan nama di kotak Nama mesin Negara.

   Untuk tutorial ini, masukkan nama **`ParentStateMachine`**.

8. (Opsional) Dalam konfigurasi mesin State, tentukan pengaturan alur kerja lainnya, seperti jenis mesin status dan peran pelaksanaannya.

   Untuk tutorial ini, simpan semua pilihan default di pengaturan mesin State.

   Jika sebelumnya Anda telah membuat peran IAM dengan izin yang benar untuk mesin status dan ingin menggunakannya, di Izin, pilih Pilih peran yang ada, lalu pilih peran dari daftar. Atau pilih Masukkan peran ARN dan kemudian berikan ARN untuk peran IAM itu.

9. Dalam kotak dialog Konfirmasi pembuatan peran, pilih Konfirmasi untuk melanjutkan.

   Anda juga dapat memilih Lihat pengaturan peran untuk kembali ke konfigurasi mesin Status.

> **Note**
>
> Jika Anda menghapus IAM role yang Step Functions buat, Step Functions tidak dapat membuatnya kembali nanti. Demikian pula, jika Anda mengubah peran (misalnya, dengan menghapus Step Functions dari principal dalam kebijakan IAM), Step Functions tidak dapat memulihkan pengaturan aslinya nanti.

Langkah 3: Perbarui kebijakan IAM

Untuk memastikan mesin status Anda memiliki izin untuk memulai eksekusi mesin status yang menggunakan fungsi Lambda, Anda harus melampirkan kebijakan inline ke peran IAM mesin status Anda. Untuk informasi selengkapnya, lihat Menyematkan kebijakan sebaris dalam Panduan Pengguna IAM.

1. Pada ParentStateMachinehalaman, pilih ARN peran IAM untuk menavigasi ke halaman Peran IAM untuk mesin status Anda.
2. Tetapkan izin yang sesuai untuk peran IAM agar dapat memulai eksekusi mesin negara lain. ParentStateMachine Untuk menetapkan izin, lakukan hal berikut:

   a. Pada halaman Peran IAM, pilih Tambahkan izin, lalu pilih Buat kebijakan sebaris.

   b. Di halaman Buat kebijakan, pilih tab JSON.

   c. Ganti teks yang ada dengan kebijakan berikut.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "states:StartExecution"
          ],
          "Resource": [
              "arn:aws:states:us-
east-2:123456789012:stateMachine:LambdaStateMachine"
          ]
      }
  ]
```

```
}
```

d. Pilih Tinjau kebijakan.

e. Tentukan nama untuk kebijakan tersebut, lalu pilih Buat kebijakan.

Langkah 4: Jalankan mesin negara

Eksekusi mesin status adalah instans tempat Anda menjalankan alur kerja untuk melakukan tugas.

1. Pada ParentStateMachinehalaman, pilih Mulai eksekusi.

   Kotak dialog Mulai eksekusi ditampilkan.

2. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

   a. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

   > **Note**
   >
   > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

   b. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

   c. Pilih Mulai Eksekusi.

   d. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

      Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

3. Buka LambdaStateMachinehalaman dan perhatikan eksekusi baru yang dipicu oleh file ParentStateMachine.

Menggunakan fungsi Lambda untuk melanjutkan eksekusi baru

Anda dapat membuat mesin status yang menggunakan fungsi Lambda untuk memulai eksekusi baru sebelum eksekusi saat ini berakhir. Menggunakan pendekatan ini untuk melanjutkan pekerjaan Anda yang sedang berlangsung dalam eksekusi baru memungkinkan Anda memiliki mesin status yang dapat memecah pekerjaan besar menjadi alur kerja yang lebih kecil, atau memiliki mesin status yang berjalan tanpa batas waktu.

Tutorial ini didasarkan pada konsep penggunaan fungsi Lambda eksternal untuk mengubah alur kerja Anda, yang ditunjukkan dalam tutorial Ulangi loop dengan Lambda. Anda menggunakan fungsi Lambda yang sama (`Iterator`) untuk mengiterasi putaran untuk beberapa jumlah tertentu. Selain itu, Anda membuat fungsi Lambda lain untuk memulai eksekusi baru dari alur kerja Anda, dan untuk mengurangi hitungan setiap kali memulai eksekusi baru. Dengan mengatur jumlah eksekusi dalam input, mesin status tersebut berakhir dan memulai eksekusi sejumlah waktu tertentu.

Mesin status yang Anda akan buat mengimplementasikan status berikut.

Status	Tujuan
`ConfigureCount`	Status `Pass` yang mengonfigurasi nilai `count`, `index`, dan `step` yang Fungsi Lambda `Iterator` gunakan untuk melangkah melalui iterasi pekerjaan.
`Iterator`	Status `Task` yang mereferensikan Fungsi Lambda `Iterator`.
`IsCountReached`	`Choice`Status yang menggunakan nilai Boolean dari `Iterator` fungsi untuk memutuskan apakah mesin status harus melanjutkan pekerjaan contoh, atau pindah ke `ShouldRestart` status.
`ExampleWork`	`Pass`Negara yang mewakili `Task` status yang akan melakukan pekerjaan dalam implementasi aktual.
`ShouldRestart`	Sebuah `Choice` negara yang menggunakan `executionCount` nilai untuk memutuskan apakah itu harus mengakhiri satu eksekusi dan memulai yang lain, atau hanya mengakhiri.
`Restart`	Status `Task` yang menggunakan fungsi Lambda untuk memulai eksekusi baru dari mesin status Anda. Seperti fungsi `Iterator`,

Status	Tujuan
	fungsi ini juga mengurangi hitungan. `RestartNegara` meneruskan nilai penghitungan yang dikurangi ke input eksekusi baru.

Prasyarat

Sebelum Anda mulai, ikuti Membuat mesin status Step Functions yang menggunakan Lambda tutorial untuk memastikan bahwa Anda terbiasa menggunakan Lambda dan Step Functions bersama-sama.

Topik

- Langkah 1: Buat fungsi Lambda untuk mengulangi hitungan
- Langkah 2: Buat fungsi Restart Lambda untuk memulai eksekusi Step Functions baru
- Langkah 3: Buat mesin negara
- Langkah 4: Perbarui Kebijakan IAM
- Langkah 5: Jalankan mesin negara

Langkah 1: Buat fungsi Lambda untuk mengulangi hitungan

> **Note**
>
> Jika Anda telah menyelesaikan tutorial Ulangi loop dengan Lambda, Anda dapat melewati langkah ini dan menggunakan fungsi Lambda.

Bagian ini dan Ulangi loop dengan Lambda tutorial menunjukkan bagaimana Anda dapat menggunakan fungsi Lambda untuk melacak hitungan, misalnya, jumlah iterasi loop di mesin status Anda.

Fungsi Lambda berikut menerima nilai input untuk `count`, `index`, dan `step`. Ia mengembalikan nilai-nilai ini dengan `index` diperbarui dan Boolean bernama `continue`. Fungsi Lambda mengatur `continue` ke `true` jika `index` kurang dari `count`.

Mesin status Anda kemudian mengimplementasikan status `Choice` yang mengeksekusi beberapa logika aplikasi jika `continue` merupakan `true`, atau pindah ke `ShouldRestart` jika `continue` adalah `false`.

Buat fungsi Lambda Iterasi

1. Buka konsol Lambda dan pilih Buat fungsi.
2. Pilih halaman Buat fungsi, pilih Penulis dari scratch.
3. Di bagian Informasi dasar, konfigurasikan fungsi Lambda Anda, sebagai berikut:

   a. Untuk Nama fungsi, masukkan `Iterator`.

   b. Untuk Runtime, pilih Node.js 16.x.

   c. Simpan semua pilihan default pada halaman, lalu pilih Buat fungsi.

   Ketika fungsi Lambda Anda dibuat, buat catatan dari Amazon Resource Name (ARN)nya di sudut kanan atas halaman, misalnya:

   ```
   arn:aws:lambda:us-east-1:123456789012:function:Iterator
   ```

4. Salin kode berikut untuk fungsi Lambda ke bagian sumber Kode halaman ***Iterator di konsol*** Lambda.

   ```
   exports.handler = function iterator (event, context, callback) {
     let index = event.iterator.index;
     let step = event.iterator.step;
     let count = event.iterator.count;

     index = index + step;

     callback(null, {
       index,
       step,
       count,
       continue: index < count
     })
   }
   ```

   Kode ini menerima nilai input untuk `count`, `index`, dan `step`. Kode ini menambah `index` dengan nilai dari `step`, dan mengembalikan nilai-nilai ini, dan nilai Boolean dari `continue`. Nilai dari `continue` adalah `true` jika `index` kurang dari `count`.

5. Pilih Deploy untuk menyebarkan kode.

Uji fungsi Iterate Lambda

Untuk melihat fungsi `Iterate` Anda bekerja, jalankan dengan nilai-nilai numerik. Anda dapat memberikan nilai input untuk fungsi Lambda yang meniru iterasi untuk melihat output yang Anda dapatkan dengan nilai input tertentu.

Untuk menguji fungsi Lambda Anda

1. Di kotak dialog Konfigurasikan peristiwa pengujian, pilih Buat peristiwa pengujian baru, lalu ketik `TestIterator` untuk Nama peristiwa.
2. Ganti contoh data dengan hal berikut.

   ```
   {
     "Comment": "Test my Iterator function",
     "iterator": {
       "count": 10,
       "index": 5,
       "step": 1
     }
   }
   ```

   Nilai-nilai ini meniru yang akan datang dari mesin status Anda selama iterasi. Fungsi Lambda menambah indeks dan mengembalikan `continue` sebagai `true`. Bila indeks tidak kurang dari `count`, ia mengembalikan `continue` sebagai `false`. Untuk tes ini, indeks telah bertambah menjadi 5. Hasil harus menambah `index` ke 6 dan atur `continue` ke `true`.
3. Pilih Buat.
4. Pada halaman ***`Iterator`*** di konsol Lambda Anda, TestIteratorpastikan terdaftar, lalu pilih Uji.

   Hasil uji ditampilkan di bagian atas halaman. Pilih Detail dan tinjau hasilnya.

   ```
   {
     "index": 6,
     "step": 1,
     "count": 10,
     "continue": true
   }
   ```

> **Note**
>
> Jika Anda mengatur `index` ke 9 untuk uji ini, `index` menaikkannya ke `10`, dan `continue` adalah `false`.

Langkah 2: Buat fungsi Restart Lambda untuk memulai eksekusi Step Functions baru

1. Buka konsol Lambda dan pilih Buat fungsi.
2. Pilih halaman Buat fungsi, pilih Penulis dari scratch.
3. Di bagian Informasi dasar, konfigurasikan fungsi Lambda Anda, sebagai berikut:
   a. Untuk Nama fungsi, masukkan `Restart`.
   b. Untuk Runtime, pilih Node.js 16.x.
4. Simpan semua pilihan default pada halaman, lalu pilih Buat fungsi.

   Ketika fungsi Lambda Anda dibuat, buat catatan dari Amazon Resource Name (ARN)nya di sudut kanan atas halaman, misalnya:

   ```
   arn:aws:lambda:us-east-1:123456789012:function:Iterator
   ```

5. Salin kode berikut untuk fungsi Lambda ke bagian Sumber kode pada halaman ***Restart di konsol*** Lambda.

   Kode berikut mengurangi hitungan jumlah eksekusi, dan memulai eksekusi baru dari mesin status Anda, termasuk nilai yang dikurangi.

   ```
   var aws = require('aws-sdk');
   var sfn = new aws.StepFunctions();

   exports.restart = function(event, context, callback) {

     let StateMachineArn = event.restart.StateMachineArn;
     event.restart.executionCount -= 1;
     event = JSON.stringify(event);

     let params = {
         input: event,
         stateMachineArn: StateMachineArn
   ```

```
  };

  sfn.startExecution(params, function(err, data) {
      if (err) callback(err);
      else callback(null,event);
  });

}
```

6. Pilih Deploy untuk menyebarkan kode.

Langkah 3: Buat mesin negara

Sekarang Anda telah membuat dua fungsi Lambda Anda, buat mesin status. Dalam mesin status ini, status `ShouldRestart` dan `Restart` adalah cara Anda memecah pekerjaan Anda menjadi beberapa eksekusi.

Example ShouldRestart Negara pilihan

Kutipan berikut menunjukkan negara. `ShouldRestart` `Choice` Status ini menentukan apakah Anda harus memulai ulang eksekusi atau tidak.

```
"ShouldRestart": {
"Type": "Choice",
"Choices": [
  {
    "Variable": "$.restart.executionCount",
    "NumericGreaterThan": 1,
    "Next": "Restart"
  }
],
```

Nilai `$.restart.executionCount` disertakan dalam input eksekusi awal. Nilai ini dikurangi satu setiap kali fungsi `Restart` dipanggil, lalu ditempatkan ke input untuk setiap eksekusi berikutnya.

Example Mulai Ulang Status Tugas

Kutipan berikut menunjukkan negara. `Restart` `Task` Status ini menggunakan fungsi Lambda yang Anda buat sebelumnya untuk memulai ulang eksekusi, dan untuk mengurangi jumlah guna melacak jumlah eksekusi yang tersisa untuk memulai.

```
"Restart": {
```

```
  "Type": "Task",
  "Resource": "arn:aws:lambda:us-east-1:123456789012:function:Restart",
  "Next": "Done"
},
```

Untuk membuat mesin negara

1. Buka Konsol Step Functions dan pilih Buat mesin status.

   > **Important**
   >
   > Pastikan mesin status Anda berada di bawah AWS akun dan Wilayah yang sama dengan fungsi Lambda yang Anda buat sebelumnya di Langkah 1 dan Langkah 2.

2. Dalam kotak dialog Pilih templat, pilih Kosong.
3. Pilih Pilih. Ini membuka Workflow Studio diMode desain.
4. Untuk tutorial ini, Anda akan menulis definisi Amazon States Language (ASL) dari mesin status Anda diEditor kode. Untuk melakukan ini, pilih Kode.
5. Hapus kode boilerplate yang ada dan tempel kode berikut. Ingatlah untuk mengganti ARN dalam kode ini dengan ARN dari fungsi Lambda yang Anda buat.

```
{
    "Comment": "Continue-as-new State Machine Example",
    "StartAt": "ConfigureCount",
    "States": {
        "ConfigureCount": {
            "Type": "Pass",
            "Result": {
                "count": 100,
                "index": -1,
                "step": 1
            },
            "ResultPath": "$.iterator",
            "Next": "Iterator"
        },
        "Iterator": {
            "Type": "Task",
            "Resource": "arn:aws:lambda:us-east-1:123456789012:function:Iterator",
            "ResultPath": "$.iterator",
            "Next": "IsCountReached"
        },
```

```
        "IsCountReached": {
            "Type": "Choice",
            "Choices": [
                {
                    "Variable": "$.iterator.continue",
                    "BooleanEquals": true,
                    "Next": "ExampleWork"
                }
            ],
            "Default": "ShouldRestart"
        },
        "ExampleWork": {
            "Comment": "Your application logic, to run a specific number of times",
            "Type": "Pass",
            "Result": {
              "success": true
            },
            "ResultPath": "$.result",
            "Next": "Iterator"
        },
        "ShouldRestart": {
          "Type": "Choice",
          "Choices": [
            {
              "Variable": "$.restart.executionCount",
              "NumericGreaterThan": 0,
              "Next": "Restart"
            }
          ],
          "Default": "Done"
        },
        "Restart": {
          "Type": "Task",
          "Resource": "arn:aws:lambda:us-east-1:123456789012:function:Restart",
          "Next": "Done"
        },
        "Done": {
            "Type": "Pass",
            "End": true
        }
    }
}
```

6. Tentukan nama untuk mesin negara Anda. Untuk melakukan ini, pilih ikon edit di sebelah nama mesin status default MyStateMachine. Kemudian, dalam konfigurasi mesin Negara, tentukan nama di kotak Nama mesin Negara.

   Untuk tutorial ini, masukkan nama **`ContinueAsNew`**.

7. (Opsional) Dalam konfigurasi mesin State, tentukan pengaturan alur kerja lainnya, seperti jenis mesin status dan peran pelaksanaannya.

   Untuk tutorial ini, simpan semua pilihan default di pengaturan mesin State.

   Jika sebelumnya Anda telah membuat peran IAM dengan izin yang benar untuk mesin status dan ingin menggunakannya, di Izin, pilih Pilih peran yang ada, lalu pilih peran dari daftar. Atau pilih Masukkan peran ARN dan kemudian berikan ARN untuk peran IAM itu.

8. Dalam kotak dialog Konfirmasi pembuatan peran, pilih Konfirmasi untuk melanjutkan.

   Anda juga dapat memilih Lihat pengaturan peran untuk kembali ke konfigurasi mesin Status.

   > **Note**
   >
   > Jika Anda menghapus IAM role yang Step Functions buat, Step Functions tidak dapat membuatnya kembali nanti. Demikian pula, jika Anda mengubah peran (misalnya, dengan menghapus Step Functions dari principal dalam kebijakan IAM), Step Functions tidak dapat memulihkan pengaturan aslinya nanti.

9. Simpan Nama Sumber Daya Amazon (ARN) dari mesin status ini dalam file teks. Anda harus memberikan ARN sambil memberikan izin ke fungsi Lambda untuk memulai eksekusi Step Functions baru.

Langkah 4: Perbarui Kebijakan IAM

Untuk memastikan fungsi Lambda Anda memiliki izin untuk memulai eksekusi Step Functions baru, lampirkan kebijakan inline ke peran IAM yang Anda gunakan untuk fungsi Lambda Anda. `Restart` Untuk informasi selengkapnya, lihat Menyematkan kebijakan sebaris dalam Panduan Pengguna IAM.

> **Note**
>
> Anda dapat memperbarui baris `Resource` dalam contoh sebelumnya untuk mereferensikan ARN mesin status `ContinueAsNew` Anda. Baris ini membatasi kebijakan sehingga hanya dapat memulai eksekusi mesin status tertentu.

```
{
 "Version": "2012-10-17",
 "Statement": [
     {
         "Sid": "VisualEditor0",
         "Effect": "Allow",
         "Action": [
             "states:StartExecution"
         ],
         "Resource": "arn:aws:states:us-east-2:123456789012stateMachine:ContinueAsNew"
     }
 ]
}
```

Langkah 5: Jalankan mesin negara

Untuk memulai eksekusi, sediakan input yang mencakup ARN dari mesin status dan `executionCount` untuk berapa kali mesin status harus memulai eksekusi baru.

1. Pada ContinueAsNewhalaman, pilih Mulai eksekusi.

   Kotak dialog Mulai eksekusi ditampilkan.
2. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

   a. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

   > **Note**
   >
   > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini

tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

b. Dalam kotak Input, masukkan input JSON berikut untuk menjalankan alur kerja Anda.

```
{
  "restart": {
    "StateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:ContinueAsNew",
    "executionCount": 4
  }
}
```

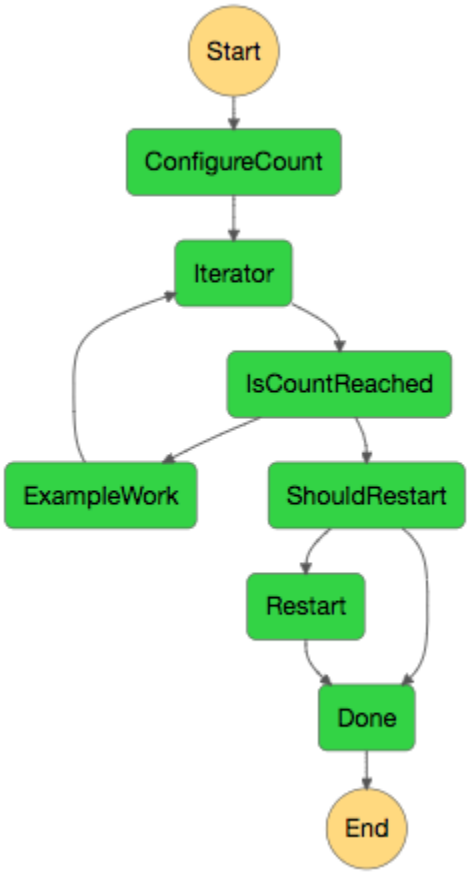
c. Perbarui bidang `StateMachineArn` dengan ARN untuk mesin status `ContinueAsNew`.

d. Pilih Mulai Eksekusi.

e. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Tampilan Grafik menampilkan yang pertama dari empat eksekusi. Sebelum selesai, ia akan melewati status `Restart` dan memulai eksekusi baru.



Saat eksekusi ini selesai, Anda dapat melihat eksekusi berikutnya yang sedang berjalan. Pilih ContinueAsNewtautan di bagian atas untuk melihat daftar eksekusi. Anda akan melihat eksekusi yang baru saja ditutup, dan eksekusi berkelanjutan yang dimulai oleh fungsi `Restart` Lambda.

Ketika semua eksekusi selesai, Anda akan melihat empat eksekusi yang berhasil dalam daftar. Eksekusi pertama yang dimulai menampilkan nama yang Anda pilih, dan eksekusi berikutnya memiliki nama yang dibuat.

Men-deploy Contoh Proyek Persetujuan Manusia

Tutorial ini menunjukkan cara men-deploy proyek persetujuan manusia yang memungkinkan eksekusi AWS Step Functions untuk dijeda selama tugas, dan menunggu pengguna untuk merespons email. Alur kerja berlanjut ke status berikutnya setelah pengguna menyetujui tugas untuk melanjutkan.

Menyebarkan AWS CloudFormation tumpukan yang disertakan dalam tutorial ini akan membuat semua sumber daya yang diperlukan, termasuk:

- Sumber daya Amazon API Gateway
- Sebuah AWS Lambda fungsi
- Mesin AWS Step Functions negara
- Topik email Amazon Simple Notification Service
- AWS Identity and Access Management Peran dan izin terkait

> **Note**
>
> Anda harus memberikan alamat email yang valid yang dapat Anda akses saat membuat AWS CloudFormation tumpukan.

Untuk informasi selengkapnya, lihat Bekerja dengan CloudFormation Template dan `AWS::StepFunctions::StateMachine` sumber daya di Panduan AWS CloudFormation Pengguna.

Topik

- Langkah 1: Buat AWS CloudFormation template
- Langkah 2: Buat tumpukan
- Langkah 3: Menyetujui langganan Amazon SNS
- Langkah 4: Jalankan mesin negara
- AWS CloudFormation Kode Sumber Template

Langkah 1: Buat AWS CloudFormation template

1. Salin kode contoh dari bagian AWS CloudFormation Kode Sumber Template.

2. Tempelkan sumber AWS CloudFormation template ke dalam file di mesin lokal Anda.

   Untuk contoh ini file disebut `human-approval.yaml`.

Langkah 2: Buat tumpukan

1. Masuk ke Konsol AWS CloudFormation.
2. Pilih Buat Tumpukan, lalu pilih Dengan sumber daya baru (standar).
3. Pada halaman Buat tumpukan, lakukan hal berikut:

   a. Di bagian Prasyarat - Siapkan templat, pastikan Template sudah siap dipilih.

   b. Di bagian Tentukan templat, pilih Unggah file templat lalu pilih Pilih file untuk mengunggah `human-approval.yaml` file yang Anda buat sebelumnya yang menyertakan kode sumber templat.

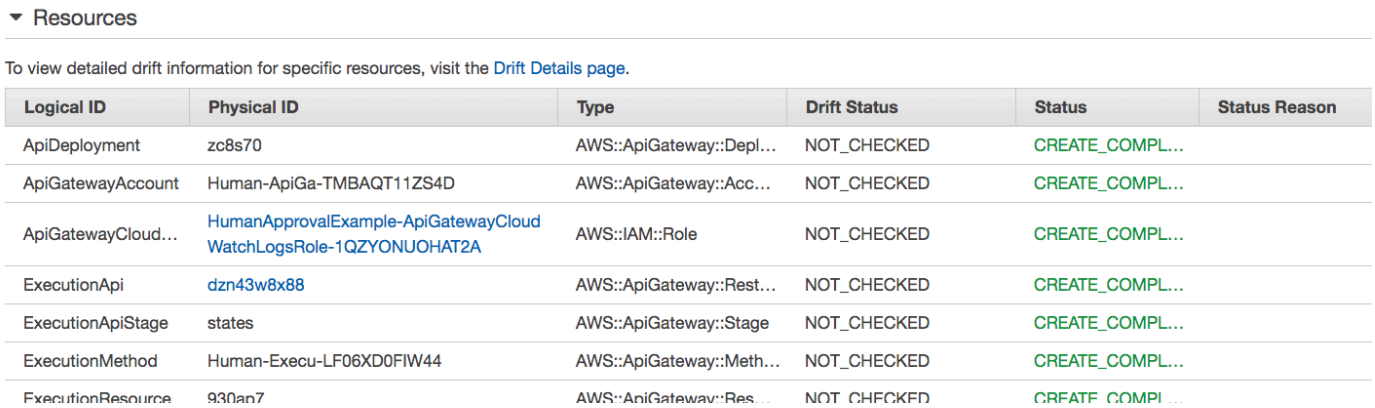
4. Pilih Berikutnya.
5. Pada halaman Tentukan detail tumpukan, lakukan hal berikut:

   a. Untuk nama Stack, masukkan nama untuk tumpukan Anda.

b. Di bawah Parameter, masukkan alamat email yang valid. Anda akan menggunakan alamat email ini untuk berlangganan topik Amazon SNS.

6. Pilih Berikutnya, lalu pilih Berikutnya lagi.
7. Pada halaman Tinjauan, pilih Saya mengakui yang AWS CloudFormation mungkin membuat sumber daya IAM dan kemudian pilih Buat.

   AWS CloudFormation mulai membuat tumpukan Anda dan menampilkan status CREATE_IN_PROGRESS. Ketika proses selesai, AWS CloudFormation menampilkan status CREATE_COMPLETE.
8. (Opsional) Untuk menampilkan sumber daya di tumpukan Anda, pilih tumpukan dan pilih tab Sumber Daya.

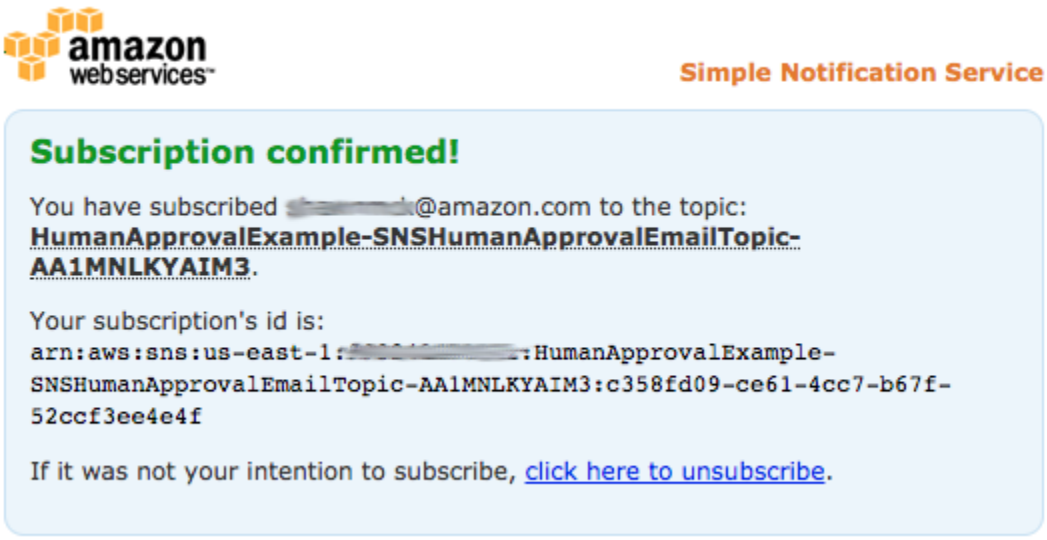


Langkah 3: Menyetujui langganan Amazon SNS

Setelah topik Amazon SNS dibuat, Anda akan menerima email yang meminta Anda mengonfirmasi langganan.

1. Buka akun email yang Anda berikan saat Anda membuat AWS CloudFormation tumpukan.
2. Buka pesan Notifikasi AWS - Konfirmasi Langganan dari no-reply@sns.amazonaws.com

   Email akan mencantumkan Amazon Resource Name untuk topik Amazon SNS, dan tautan konfirmasi.
3. Pilih tautan konfirmasikan langganan.



Langkah 4: Jalankan mesin negara

1. Pada HumanApprovalLambdaStateMachinehalaman, pilih Mulai eksekusi.

   Kotak dialog Mulai eksekusi ditampilkan.

2. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

   a. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

   > **Note**
   >
   > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

   b. Dalam kotak Input, masukkan input JSON berikut untuk menjalankan alur kerja Anda.

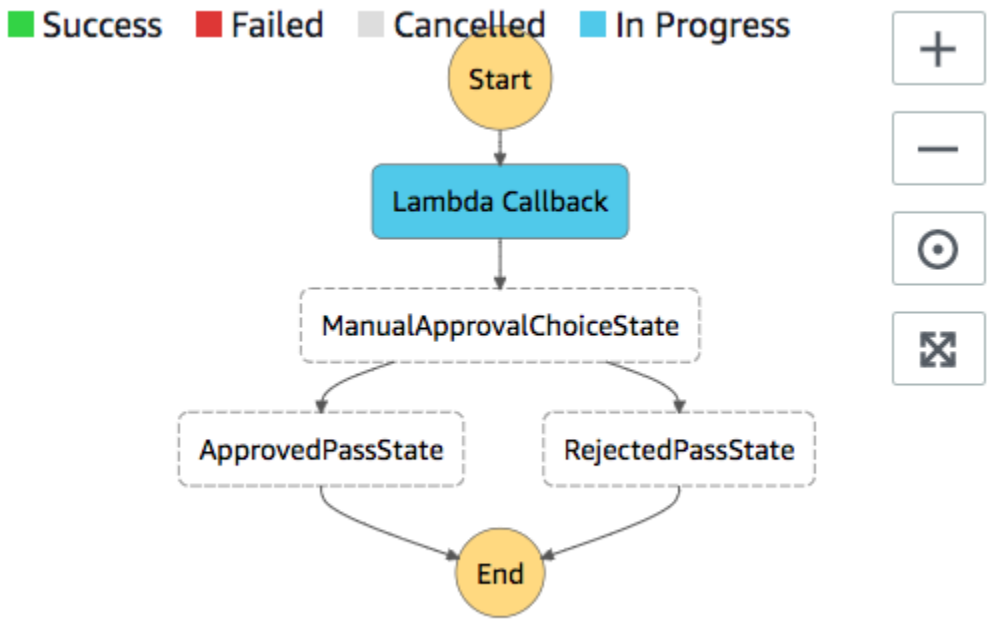
```
{
    "Comment": "Testing the human approval tutorial."
}
```

c. Pilih Mulai Eksekusi.

Eksekusi mesin ApprovalTeststatus dimulai, dan berhenti pada tugas Lambda Callback.

d. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

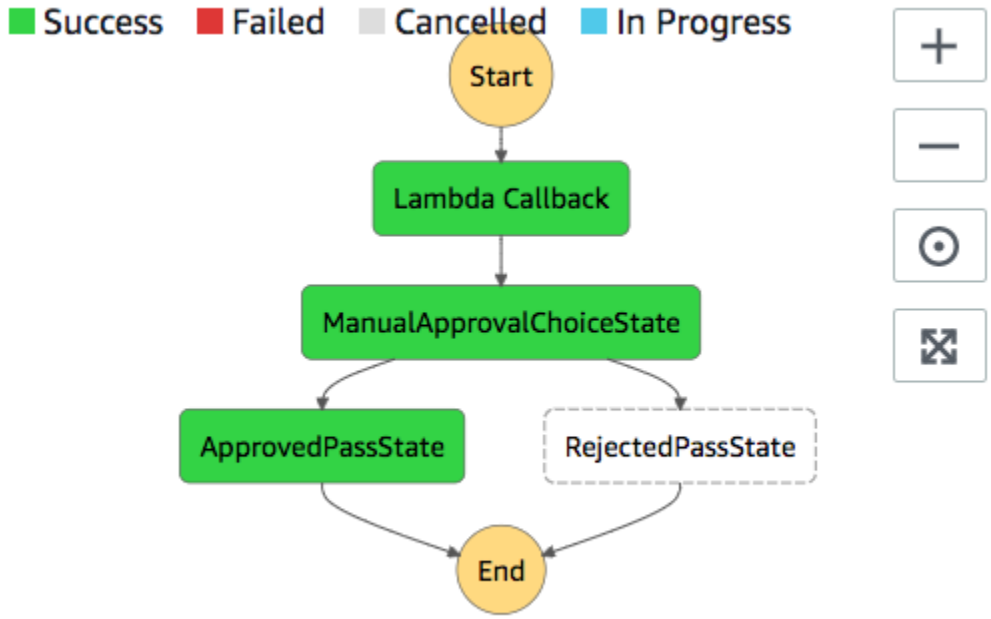


3. Di akun email yang Anda gunakan untuk topik Amazon SNS sebelumnya, buka pesan dengan subjek Persetujuan yang diperlukan dari. AWS Step Functions

Pesan mencakup URL terpisah untuk Setujui dan Tolak.

4. Pilih Setujui URL.

Alur kerja berlanjut berdasarkan pilihan Anda.



AWS CloudFormation Kode Sumber Template

Gunakan AWS CloudFormation template ini untuk menyebarkan contoh alur kerja proses persetujuan manusia.

```
AWSTemplateFormatVersion: "2010-09-09"
Description: "AWS Step Functions Human based task example. It sends an email with an
 HTTP URL for approval."
Parameters:
  Email:
    Type: String
    AllowedPattern: "^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\\.[a-zA-Z0-9-.]+$"
    ConstraintDescription: Must be a valid email address.
Resources:
  # Begin API Gateway Resources
  ExecutionApi:
    Type: "AWS::ApiGateway::RestApi"
    Properties:
      Name: "Human approval endpoint"
      Description: "HTTP Endpoint backed by API Gateway and Lambda"
      FailOnWarnings: true

  ExecutionResource:
```

```
    Type: 'AWS::ApiGateway::Resource'
    Properties:
      RestApiId: !Ref ExecutionApi
      ParentId: !GetAtt "ExecutionApi.RootResourceId"
      PathPart: execution

  ExecutionMethod:
    Type: "AWS::ApiGateway::Method"
    Properties:
      AuthorizationType: NONE
      HttpMethod: GET
      Integration:
        Type: AWS
        IntegrationHttpMethod: POST
        Uri: !Sub "arn:aws:apigateway:${AWS::Region}:lambda:path/2015-03-31/functions/
${LambdaApprovalFunction.Arn}/invocations"
        IntegrationResponses:
          - StatusCode: 302
            ResponseParameters:
              method.response.header.Location:
 "integration.response.body.headers.Location"
        RequestTemplates:
          application/json: |
            {
              "body" : $input.json('$'),
              "headers": {
                #foreach($header in $input.params().header.keySet())
                "$header":
 "$util.escapeJavaScript($input.params().header.get($header))"
 #if($foreach.hasNext),#end

                #end
              },
              "method": "$context.httpMethod",
              "params": {
                #foreach($param in $input.params().path.keySet())
                "$param": "$util.escapeJavaScript($input.params().path.get($param))"
 #if($foreach.hasNext),#end

                #end
              },
              "query": {
                #foreach($queryParam in $input.params().querystring.keySet())
```

```
                "$queryParam":
"$util.escapeJavaScript($input.params().querystring.get($queryParam))"
#if($foreach.hasNext),#end

                #end
              }
            }
      ResourceId: !Ref ExecutionResource
      RestApiId: !Ref ExecutionApi
      MethodResponses:
        - StatusCode: 302
          ResponseParameters:
            method.response.header.Location: true

  ApiGatewayAccount:
    Type: 'AWS::ApiGateway::Account'
    Properties:
      CloudWatchRoleArn: !GetAtt "ApiGatewayCloudWatchLogsRole.Arn"

  ApiGatewayCloudWatchLogsRole:
    Type: 'AWS::IAM::Role'
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - apigateway.amazonaws.com
            Action:
              - 'sts:AssumeRole'
      Policies:
        - PolicyName: ApiGatewayLogsPolicy
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                  - "logs:*"
                Resource: !Sub "arn:${AWS::Partition}:logs:*:*:*"

  ExecutionApiStage:
    DependsOn:
      - ApiGatewayAccount
```

```
    Type: 'AWS::ApiGateway::Stage'
    Properties:
      DeploymentId: !Ref ApiDeployment
      MethodSettings:
        - DataTraceEnabled: true
          HttpMethod: '*'
          LoggingLevel: INFO
          ResourcePath: /*
      RestApiId: !Ref ExecutionApi
      StageName: states

  ApiDeployment:
    Type: "AWS::ApiGateway::Deployment"
    DependsOn:
      - ExecutionMethod
    Properties:
      RestApiId: !Ref ExecutionApi
      StageName: DummyStage
  # End API Gateway Resources

  # Begin
  # Lambda that will be invoked by API Gateway
  LambdaApprovalFunction:
    Type: 'AWS::Lambda::Function'
    Properties:
      Code:
        ZipFile:
          Fn::Sub: |
            const { SFN: StepFunctions } = require("@aws-sdk/client-sfn");
            var redirectToStepFunctions = function(lambdaArn, statemachineName,
executionName, callback) {
              const lambdaArnTokens = lambdaArn.split(":");
              const partition = lambdaArnTokens[1];
              const region = lambdaArnTokens[3];
              const accountId = lambdaArnTokens[4];

              console.log("partition=" + partition);
              console.log("region=" + region);
              console.log("accountId=" + accountId);

              const executionArn = "arn:" + partition + ":states:" + region + ":" +
accountId + ":execution:" + statemachineName + ":" + executionName;
              console.log("executionArn=" + executionArn);
```

```
            const url = "https://console.aws.amazon.com/states/home?region=" + region
+ "#/executions/details/" + executionArn;
            callback(null, {
                statusCode: 302,
                headers: {
                  Location: url
                }
            });
          };

          exports.handler = (event, context, callback) => {
            console.log('Event= ' + JSON.stringify(event));
            const action = event.query.action;
            const taskToken = event.query.taskToken;
            const statemachineName = event.query.sm;
            const executionName = event.query.ex;

            const stepfunctions = new StepFunctions();

            var message = "";

            if (action === "approve") {
              message = { "Status": "Approved! Task approved by ${Email}" };
            } else if (action === "reject") {
              message = { "Status": "Rejected! Task rejected by ${Email}" };
            } else {
              console.error("Unrecognized action. Expected: approve, reject.");
              callback({"Status": "Failed to process the request. Unrecognized
Action."});
            }

            stepfunctions.sendTaskSuccess({
              output: JSON.stringify(message),
              taskToken: event.query.taskToken
            })
            .then(function(data) {
              redirectToStepFunctions(context.invokedFunctionArn, statemachineName,
executionName, callback);
            }).catch(function(err) {
              console.error(err, err.stack);
              callback(err);
            });
          }
     Description: Lambda function that callback to AWS Step Functions
```

```
      FunctionName: LambdaApprovalFunction
      Handler: index.handler
      Role: !GetAtt "LambdaApiGatewayIAMRole.Arn"
      Runtime: nodejs18.x

  LambdaApiGatewayInvoke:
    Type: "AWS::Lambda::Permission"
    Properties:
      Action: "lambda:InvokeFunction"
      FunctionName: !GetAtt "LambdaApprovalFunction.Arn"
      Principal: "apigateway.amazonaws.com"
      SourceArn: !Sub "arn:aws:execute-api:${AWS::Region}:${AWS::AccountId}:
${ExecutionApi}/*"

  LambdaApiGatewayIAMRole:
    Type: "AWS::IAM::Role"
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Action:
              - "sts:AssumeRole"
            Effect: "Allow"
            Principal:
              Service:
                - "lambda.amazonaws.com"
      Policies:
        - PolicyName: CloudWatchLogsPolicy
          PolicyDocument:
            Statement:
              - Effect: Allow
                Action:
                  - "logs:*"
                Resource: !Sub "arn:${AWS::Partition}:logs:*:*:*"
        - PolicyName: StepFunctionsPolicy
          PolicyDocument:
            Statement:
              - Effect: Allow
                Action:
                  - "states:SendTaskFailure"
                  - "states:SendTaskSuccess"
                Resource: "*"
  # End Lambda that will be invoked by API Gateway
```

```
  # Begin state machine that publishes to Lambda and sends an email with the link for
 approval
  HumanApprovalLambdaStateMachine:
    Type: AWS::StepFunctions::StateMachine
    Properties:
      RoleArn: !GetAtt LambdaStateMachineExecutionRole.Arn
      DefinitionString:
        Fn::Sub: |
          {
              "StartAt": "Lambda Callback",
              "TimeoutSeconds": 3600,
              "States": {
                  "Lambda Callback": {
                      "Type": "Task",
                      "Resource": "arn:
${AWS::Partition}:states:::lambda:invoke.waitForTaskToken",
                      "Parameters": {
                        "FunctionName": "${LambdaHumanApprovalSendEmailFunction.Arn}",
                        "Payload": {
                          "ExecutionContext.$": "$$",
                          "APIGatewayEndpoint": "https://${ExecutionApi}.execute-api.
${AWS::Region}.amazonaws.com/states"
                        }
                      },
                      "Next": "ManualApprovalChoiceState"
                  },
                  "ManualApprovalChoiceState": {
                    "Type": "Choice",
                    "Choices": [
                      {
                        "Variable": "$.Status",
                        "StringEquals": "Approved! Task approved by ${Email}",
                        "Next": "ApprovedPassState"
                      },
                      {
                        "Variable": "$.Status",
                        "StringEquals": "Rejected! Task rejected by ${Email}",
                        "Next": "RejectedPassState"
                      }
                    ]
                  },
                  "ApprovedPassState": {
                    "Type": "Pass",
                    "End": true
```

```
                },
                "RejectedPassState": {
                  "Type": "Pass",
                  "End": true
                }
            }
        }

  SNSHumanApprovalEmailTopic:
    Type: AWS::SNS::Topic
    Properties:
      Subscription:
        -
          Endpoint: !Sub ${Email}
          Protocol: email

  LambdaHumanApprovalSendEmailFunction:
    Type: "AWS::Lambda::Function"
    Properties:
      Handler: "index.lambda_handler"
      Role: !GetAtt LambdaSendEmailExecutionRole.Arn
      Runtime: "nodejs18.x"
      Timeout: "25"
      Code:
        ZipFile:
          Fn::Sub: |
            console.log('Loading function');
            const { SNS } = require("@aws-sdk/client-sns");
            exports.lambda_handler = (event, context, callback) => {
                console.log('event= ' + JSON.stringify(event));
                console.log('context= ' + JSON.stringify(context));

                const executionContext = event.ExecutionContext;
                console.log('executionContext= ' + executionContext);

                const executionName = executionContext.Execution.Name;
                console.log('executionName= ' + executionName);

                const statemachineName = executionContext.StateMachine.Name;
                console.log('statemachineName= ' + statemachineName);

                const taskToken = executionContext.Task.Token;
                console.log('taskToken= ' + taskToken);
```

```
                const apigwEndpint = event.APIGatewayEndpoint;
                console.log('apigwEndpint = ' + apigwEndpint)

                const approveEndpoint = apigwEndpint + "/execution?
action=approve&ex=" + executionName + "&sm=" + statemachineName + "&taskToken=" +
 encodeURIComponent(taskToken);
                console.log('approveEndpoint= ' + approveEndpoint);

                const rejectEndpoint = apigwEndpint + "/execution?
action=reject&ex=" + executionName + "&sm=" + statemachineName + "&taskToken=" +
 encodeURIComponent(taskToken);
                console.log('rejectEndpoint= ' + rejectEndpoint);

                const emailSnsTopic = "${SNSHumanApprovalEmailTopic}";
                console.log('emailSnsTopic= ' + emailSnsTopic);

                var emailMessage = 'Welcome! \n\n';
                emailMessage += 'This is an email requiring an approval for a step
 functions execution. \n\n'
                emailMessage += 'Please check the following information and click
 "Approve" link if you want to approve. \n\n'
                emailMessage += 'Execution Name -> ' + executionName + '\n\n'
                emailMessage += 'Approve ' + approveEndpoint + '\n\n'
                emailMessage += 'Reject ' + rejectEndpoint + '\n\n'
                emailMessage += 'Thanks for using Step functions!'

                const sns = new SNS();
                var params = {
                  Message: emailMessage,
                  Subject: "Required approval from AWS Step Functions",
                  TopicArn: emailSnsTopic
                };

                sns.publish(params)
                  .then(function(data) {
                    console.log("MessageID is " + data.MessageId);
                    callback(null);
                  }).catch(
                    function(err) {
                    console.error(err, err.stack);
                    callback(err);
                  });
            }
```

```
  LambdaStateMachineExecutionRole:
    Type: "AWS::IAM::Role"
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Principal:
              Service: states.amazonaws.com
            Action: "sts:AssumeRole"
      Policies:
        - PolicyName: InvokeCallbackLambda
          PolicyDocument:
            Statement:
              - Effect: Allow
                Action:
                  - "lambda:InvokeFunction"
                Resource:
                  - !Sub "${LambdaHumanApprovalSendEmailFunction.Arn}"

  LambdaSendEmailExecutionRole:
    Type: "AWS::IAM::Role"
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Principal:
              Service: lambda.amazonaws.com
            Action: "sts:AssumeRole"
      Policies:
        - PolicyName: CloudWatchLogsPolicy
          PolicyDocument:
            Statement:
              - Effect: Allow
                Action:
                  - "logs:CreateLogGroup"
                  - "logs:CreateLogStream"
                  - "logs:PutLogEvents"
                Resource: !Sub "arn:${AWS::Partition}:logs:*:*:*"
        - PolicyName: SNSSendEmailPolicy
          PolicyDocument:
            Statement:
              - Effect: Allow
```

```
                Action:
                  - "SNS:Publish"
                Resource:
                  - !Sub "${SNSHumanApprovalEmailTopic}"

# End state machine that publishes to Lambda and sends an email with the link for
 approval
Outputs:
  ApiGatewayInvokeURL:
    Value: !Sub "https://${ExecutionApi}.execute-api.${AWS::Region}.amazonaws.com/
states"
  StateMachineHumanApprovalArn:
    Value: !Ref HumanApprovalLambdaStateMachine
```

Lihat jejak X-Ray di Step Functions

Dalam tutorial ini, Anda akan belajar cara menggunakan X-Ray untuk melacak kesalahan yang terjadi saat menjalankan mesin status. Anda dapat menggunakan AWS X-Ray untuk memvisualisasikan komponen mesin status Anda, mengidentifikasi hambatan performa, dan memecahkan masalah permintaan yang mengakibatkan kesalahan. Dalam tutorial ini, Anda akan membuat beberapa fungsi Lambda yang secara acak menghasilkan kesalahan, yang kemudian dapat Anda lacak dan analisis menggunakan X-Ray.

Tutorial Membuat mesin status Step Functions yang menggunakan Lambda memandu Anda dalam membuat mesin status yang memanggil fungsi Lambda. Jika Anda telah menyelesaikan tutorial tersebut, lewati ke Langkah 2 dan gunakan AWS Identity and Access Management (IAM) role yang sebelumnya Anda buat.

Topik

- Langkah 1: Buat peran IAM untuk Lambda
- Langkah 2: Buat fungsi Lambda
- Langkah 3: Buat dua fungsi Lambda lagi
- Langkah 4: Buat mesin negara
- Langkah 5: Jalankan mesin negara

Langkah 1: Buat peran IAM untuk Lambda

Keduanya AWS Lambda dan AWS Step Functions dapat mengeksekusi kode dan mengakses AWS sumber daya (misalnya, data yang disimpan di bucket Amazon S3). Untuk menjaga keamanan, Anda harus memberikan akses kepada Lambda dan Step Functions ke sumber daya ini.

Lambda mengharuskan Anda untuk menetapkan peran AWS Identity and Access Management (IAM) saat membuat fungsi Lambda, dengan cara yang sama Step Functions mengharuskan Anda menetapkan peran IAM saat membuat mesin status.

Anda dapat menggunakan konsol IAM untuk membuat peran tertaut layanan.

Untuk membuat peran (konsol)

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi konsol IAM, pilih Peran. Lalu pilih Buat peran.
3. Pilih tipe peran Layanan AWS , lalu pilih Lambda.
4. Pilih kasus penggunaan Lambda. Kasus penggunaan ditentukan oleh layanan untuk menyertakan kebijakan kepercayaan yang disyaratkan oleh layanan. Kemudian pilih Selanjutnya: Izin.
5. Pilih satu atau beberapa kebijakan izin untuk dilampirkan ke peran tersebut (misalnya, `AWSLambdaBasicExecutionRole`). Lihat Model izin AWS Lambda.

   Pilih kotak di samping kebijakan yang menetapkan izin yang Anda inginkan untuk peran tersebut, lalu pilih Selanjutnya: Tinjauan.
6. Masukkan Nama peran.
7. (Opsional) Untuk Deskripsi peran, edit deskripsi untuk peran tertaut layanan baru.
8. Tinjau peran lalu pilih Buat peran.

Langkah 2: Buat fungsi Lambda

Fungsi Lambda Anda akan secara acak membuang kesalahan atau timeout, menghasilkan contoh data untuk dilihat di X-Ray.

> **Important**
>
> Pastikan fungsi Lambda Anda berada di bawah AWS akun dan AWS Wilayah yang sama dengan mesin negara bagian Anda.

1. Buka konsol Lambda dan pilih Create Function.
2. Di bagian Buat fungsi, pilih Tulis dari awal.
3. Di bagian Informasi dasar, konfigurasikan fungsi Lambda Anda:
   a. Untuk Nama fungsi, masukkan `TestFunction1`.
   b. Untuk Runtime, pilih Node.js 18.x.
   c. Untuk Peran, pilih Pilih peran yang sudah ada.
   d. Untuk Peran yang sudah ada, pilih Peran Lambda yang Anda buat sebelumnya.

      > **Note**
      >
      > Jika IAM role yang Anda buat tidak muncul dalam daftar, peran tersebut mungkin masih memerlukan beberapa menit untuk diterapkan ke Lambda.

   e. Pilih Buat fungsi.

      Saat fungsi Lambda Anda dibuat, catat Amazon Resource Name (ARN) di sudut kanan atas halaman. Sebagai contoh:

      ```
      arn:aws:lambda:us-east-1:123456789012:function:TestFunction1
      ```

4. Salin kode berikut untuk fungsi Lambda ke bagian kode Fungsi dari halaman ***TestFunction1***.

```
function getRandomSeconds(max) {
    return Math.floor(Math.random() * Math.floor(max)) * 1000;
}
function sleep(ms) {
    return new Promise(resolve => setTimeout(resolve, ms));
}
export const handler = async (event) => {
    if(getRandomSeconds(4) === 0) {
        throw new Error("Something went wrong!");
    }
```

```
    let wait_time = getRandomSeconds(5);
    await sleep(wait_time);
    return { 'response': true }
};
```

Kode ini membuat kegagalan dengan waktu acak, yang akan digunakan untuk membuat contoh kesalahan di mesin status Anda yang dapat dilihat dan dianalisis menggunakan pelacakan X-Ray.

5. Pilih Simpan.

Langkah 3: Buat dua fungsi Lambda lagi

Buat dua fungsi Lambda lagi.

1. Ulangi Langkah 2 untuk membuat dua fungsi Lambda lagi. Untuk fungsi selanjutnya, di Nama fungsi, masukkan `TestFunction2`. Untuk fungsi terakhir, di Nama fungsi, masukkan `TestFunction3`.
2. Di konsol Lambda, periksa bahwa Anda sekarang memiliki tiga fungsi Lambda, `TestFunction1`, `TestFunction2`, dan `TestFunction3`.

Langkah 4: Buat mesin negara

Pada langkah ini, Anda akan menggunakan konsol Step Functions untuk membuat mesin status dengan tiga `Task` status. Setiap status `Task` akan mereferensikan salah satu dari tiga fungsi Lambda Anda.

1. Buka Konsol Step Functions dan pilih Buat mesin status.

> **Important**
>
> Pastikan mesin status Anda berada di bawah AWS akun dan Wilayah yang sama dengan fungsi Lambda yang Anda buat sebelumnya di Langkah 2 dan Langkah 3.

2. Dalam kotak dialog Pilih templat, pilih Kosong.
3. Pilih Pilih. Ini membuka Workflow Studio diMode desain.
4. Untuk tutorial ini, Anda akan menulis definisi Amazon States Language (ASL) dari mesin status Anda diEditor kode. Untuk melakukan ini, pilih Kode.

5. Hapus kode boilerplate yang ada dan tempel kode berikut. Dalam definisi status Tugas, ingatlah untuk mengganti contoh ARN dengan ARN dari fungsi Lambda yang Anda buat.

```
{
  "StartAt": "CallTestFunction1",
  "States": {
    "CallTestFunction1": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:test-function1",
      "Catch": [
        {
          "ErrorEquals": [
            "States.TaskFailed"
          ],
          "Next": "AfterTaskFailed"
        }
      ],
      "Next": "CallTestFunction2"
    },
    "CallTestFunction2": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:test-function2",
      "Catch": [
        {
          "ErrorEquals": [
            "States.TaskFailed"
          ],
          "Next": "AfterTaskFailed"
        }
      ],
      "Next": "CallTestFunction3"
    },
    "CallTestFunction3": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:test-function3",
      "TimeoutSeconds": 5,
      "Catch": [
        {
          "ErrorEquals": [
            "States.Timeout"
          ],
          "Next": "AfterTimeout"
        },
```

```
        {
          "ErrorEquals": [
            "States.TaskFailed"
          ],
          "Next": "AfterTaskFailed"
        }
      ],
      "Next": "Succeed"
    },
    "Succeed": {
      "Type": "Succeed"
    },
    "AfterTimeout": {
      "Type": "Fail"
    },
    "AfterTaskFailed": {
      "Type": "Fail"
    }
  }
}
```

ARN ini merupakan deskripsi dari mesin status Anda menggunakan Bahasa Status Amazon. ARN ini menentukan tiga status `Task` bernama `CallTestFunction1`, `CallTestFunction2` dan `CallTestFunction3`. Masing-masing memanggil salah satu dari tiga fungsi Lambda Anda. Untuk informasi selengkapnya, lihat Struktur Mesin Status.

6. Tentukan nama untuk mesin negara Anda. Untuk melakukan ini, pilih ikon edit di sebelah nama mesin status default MyStateMachine. Kemudian, dalam konfigurasi mesin Negara, tentukan nama di kotak Nama mesin Negara.

   Untuk tutorial ini, masukkan nama **`TraceFunctions`**.

7. (Opsional) Dalam konfigurasi mesin State, tentukan pengaturan alur kerja lainnya, seperti jenis mesin status dan peran pelaksanaannya.

   Untuk tutorial ini, di bawah Konfigurasi tambahan, pilih Aktifkan penelusuran X-Ray. Simpan semua pilihan default lainnya di pengaturan mesin State.

   Jika sebelumnya Anda telah membuat peran IAM dengan izin yang benar untuk mesin status dan ingin menggunakannya, di Izin, pilih Pilih peran yang ada, lalu pilih peran dari daftar. Atau pilih Masukkan peran ARN dan kemudian berikan ARN untuk peran IAM itu.

8. Dalam kotak dialog Konfirmasi pembuatan peran, pilih Konfirmasi untuk melanjutkan.

Anda juga dapat memilih Lihat pengaturan peran untuk kembali ke konfigurasi mesin Status.

> **Note**
>
> Jika Anda menghapus IAM role yang Step Functions buat, Step Functions tidak dapat membuatnya kembali nanti. Demikian pula, jika Anda mengubah peran (misalnya, dengan menghapus Step Functions dari principal dalam kebijakan IAM), Step Functions tidak dapat memulihkan pengaturan aslinya nanti.

Langkah 5: Jalankan mesin negara

Eksekusi mesin status adalah instans tempat Anda menjalankan alur kerja untuk melakukan tugas.

1. Pada ***`TraceFunctions`***halaman, pilih Mulai eksekusi.

   Halaman Eksekusi baru ditampilkan.
2. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

   a. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

   > **Note**
   >
   > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

   b. Pilih Mulai Eksekusi.

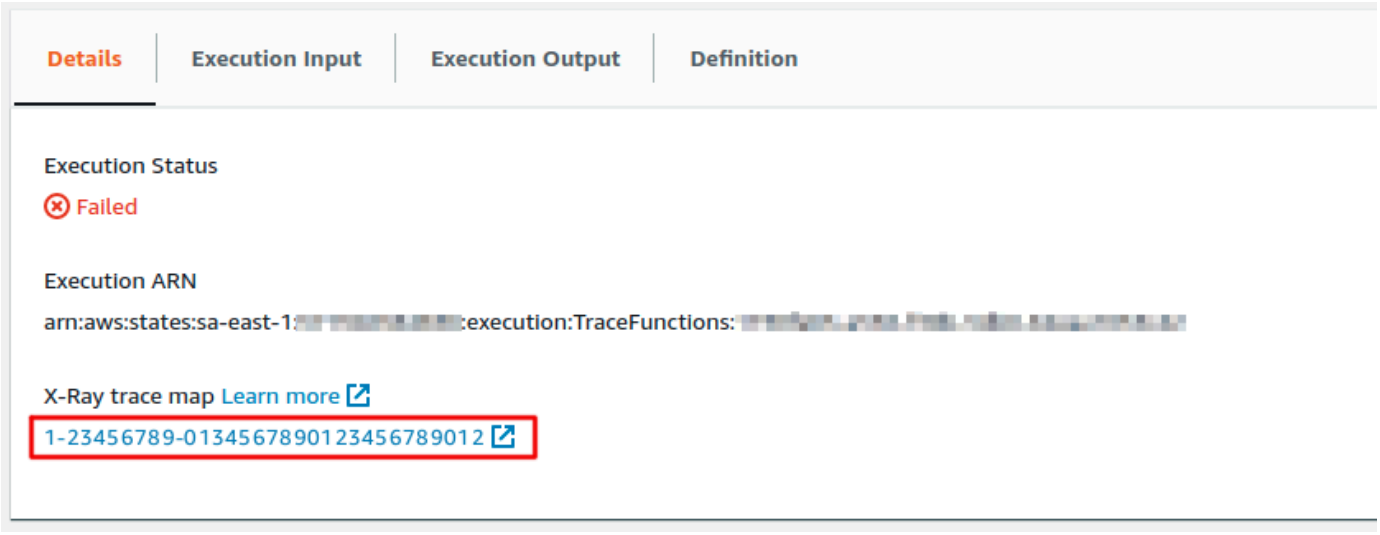
   c. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

   Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output,

dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Jalankan beberapa (setidaknya tiga) eksekusi.

3. Setelah eksekusi selesai, ikuti tautan Peta jejak X-Ray. Anda dapat melihat jejak saat eksekusi masih berjalan, tetapi Anda mungkin ingin melihat hasil eksekusi sebelum melihat peta jejak X-Ray.



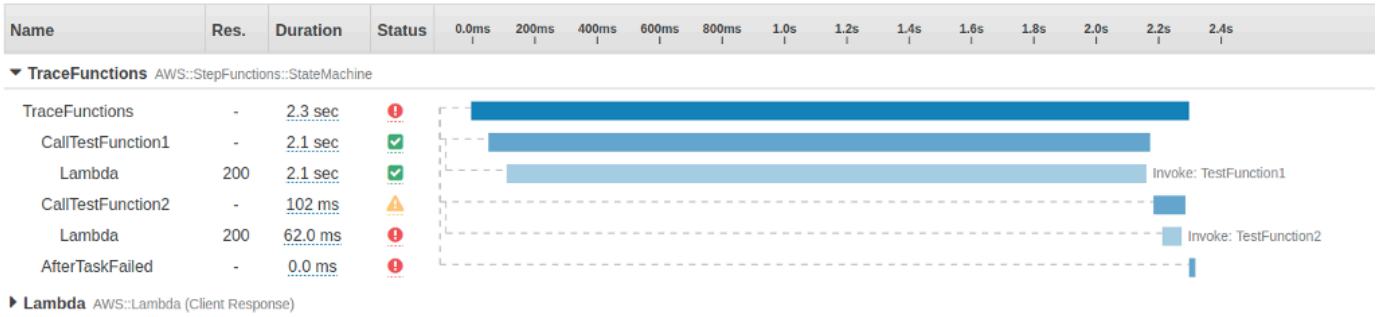
4. Lihat peta layanan untuk mengidentifikasi tempat terjadinya kesalahan, koneksi dengan latensi tinggi, atau pelacakan permintaan yang tidak berhasil. Dalam contoh ini, Anda dapat melihat banyaknya lalu lintas yang diterima oleh setiap fungsi. `TestFunction2` lebih sering dipanggil daripada `TestFunction3`, dan `TestFunction1` dipanggil lebih dari dua kali lebih sering dari `TestFunction2`.

   Peta layanan menunjukkan kondisi setiap simpul dengan mewarnainya berdasarkan rasio panggilan yang berhasil terhadap kesalahan dan kerusakan:

   - Hijau untuk panggilan berhasil
   - Merah untuk kesalahan server (500 kesalahan seri)
   - Kuning untuk kesalahan klien (400 kesalahan seri)
   - Ungu untuk kesalahan throttling (429 Terlalu Banyak Permintaan)

Anda juga dapat memilih simpul layanan untuk melihat permintaan untuk simpul tersebut, atau tepi antara dua simpul untuk melihat permintaan yang melewati koneksi tersebut.

5. Lihat peta jejak X-Ray untuk melihat peristiwa yang telah terjadi untuk setiap eksekusi. Tampilan Linimasa menunjukkan hierarki segmen dan subsegmen. Entri pertama dalam daftar adalah segmen, yang mewakili semua data yang dicatat oleh layanan untuk satu permintaan. Di bawah segmen adalah subsegmen. Contoh ini menunjukkan subsegmen yang dicatat oleh fungsi Lambda.



Untuk informasi selengkapnya tentang memahami jejak X-Ray dan menggunakan X-Ray dengan Step Functions, lihat AWS X-Ray dan Step Functions

Kumpulkan info bucket Amazon S3 menggunakan integrasi layanan AWS SDK

Tutorial ini menunjukkan cara melakukan integrasi AWS SDK dengan Amazon Simple Storage Service. Mesin status yang Anda buat dalam tutorial ini mengumpulkan informasi tentang bucket Amazon S3 Anda, lalu daftar bucket Anda bersama dengan informasi versi untuk setiap bucket di wilayah saat ini.

Topik

- Langkah 1: Buat mesin negara
- Langkah 2: Tambahkan izin peran IAM yang diperlukan
- Langkah 3: Jalankan eksekusi mesin status Standar
- Langkah 4: Jalankan eksekusi mesin status Express

Langkah 1: Buat mesin negara

Menggunakan konsol Step Functions, Anda akan membuat mesin status yang menyertakan `Task` status untuk mencantumkan semua bucket Amazon S3 di akun dan wilayah saat ini. Kemudian, Anda akan menambahkan `Task` status lain yang memanggil `HeadBucket` API untuk memverifikasi apakah bucket yang dikembalikan dapat diakses di wilayah saat ini. Jika bucket tidak dapat diakses, panggilan `HeadBucket` API mengembalikan `S3.S3Exception` kesalahan. Anda akan menyertakan `Catch` blok untuk menangkap pengecualian ini dan `Pass` status sebagai status fallback.

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Dalam kotak dialog Pilih templat, pilih Kosong.
3. Pilih Pilih. Ini membuka Workflow Studio diMode desain.
4. Untuk tutorial ini, Anda akan menulis definisi Amazon States Language (ASL) dari mesin status Anda diEditor kode. Untuk melakukan ini, pilih Kode.
5. Hapus kode boilerplate yang ada dan tempel definisi mesin status berikut.

```
{
  "Comment": "A description of my state machine",
  "StartAt": "ListBuckets",
  "States": {
    "ListBuckets": {
      "Type": "Task",
```

```
      "Parameters": {},
      "Resource": "arn:aws:states:::aws-sdk:s3:listBuckets",
      "Next": "Map"
    },
    "Map": {
      "Type": "Map",
      "ItemsPath": "$.Buckets",
      "ItemProcessor": {
        "ProcessorConfig": {
          "Mode": "INLINE"
        },
        "StartAt": "HeadBucket",
        "States": {
          "HeadBucket": {
            "Type": "Task",
            "ResultPath": null,
            "Parameters": {
              "Bucket.$": "$.Name"
            },
            "Resource": "arn:aws:states:::aws-sdk:s3:headBucket",
            "Catch": [
              {
                "ErrorEquals": [
                  "S3.S3Exception"
                ],
                "ResultPath": null,
                "Next": "Pass"
              }
            ],
            "Next": "GetBucketVersioning"
          },
          "GetBucketVersioning": {
            "Type": "Task",
            "End": true,
            "Parameters": {
              "Bucket.$": "$.Name"
            },
            "ResultPath": "$.BucketVersioningInfo",
            "Resource": "arn:aws:states:::aws-sdk:s3:getBucketVersioning"
          },
          "Pass": {
            "Type": "Pass",
            "End": true,
            "Result": {
```

```
              "Status": "Unknown"
            },
            "ResultPath": "$.BucketVersioningInfo"
          }
        }
      },
      "End": true
    }
  }
}
```

6. Tentukan nama untuk mesin negara Anda. Untuk melakukan ini, pilih ikon edit di sebelah nama mesin status default MyStateMachine. Kemudian, dalam konfigurasi mesin Negara, tentukan nama di kotak Nama mesin Negara.

   Untuk tutorial ini, masukkan nama **`Gather-S3-Bucket-Info-Standard`**.

7. (Opsional) Dalam konfigurasi mesin State, tentukan pengaturan alur kerja lainnya, seperti jenis mesin status dan peran pelaksanaannya.

   Simpan semua pilihan default dalam pengaturan mesin Negara.

   Jika sebelumnya Anda telah membuat peran IAM dengan izin yang benar untuk mesin status dan ingin menggunakannya, di Izin, pilih Pilih peran yang ada, lalu pilih peran dari daftar. Atau pilih Masukkan peran ARN dan kemudian berikan ARN untuk peran IAM itu.

8. Dalam kotak dialog Konfirmasi pembuatan peran, pilih Konfirmasi untuk melanjutkan.

   Anda juga dapat memilih Lihat pengaturan peran untuk kembali ke konfigurasi mesin Status.

   > **Note**
   >
   > Jika Anda menghapus IAM role yang Step Functions buat, Step Functions tidak dapat membuatnya kembali nanti. Demikian pula, jika Anda mengubah peran (misalnya, dengan menghapus Step Functions dari principal dalam kebijakan IAM), Step Functions tidak dapat memulihkan pengaturan aslinya nanti.

   Pada Langkah 2, Anda akan menambahkan izin yang hilang ke peran mesin status.

Langkah 2: Tambahkan izin peran IAM yang diperlukan

Untuk mengumpulkan informasi tentang bucket Amazon S3 di wilayah Anda saat ini, Anda harus memberikan izin yang diperlukan mesin status Anda untuk mengakses bucket Amazon S3.

1. Pada halaman mesin status, pilih ARN peran IAM untuk membuka halaman Peran untuk peran mesin status.
2. Pilih Tambahkan izin, lalu pilih Buat kebijakan sebaris.
3. Pilih tab JSON, lalu tempelkan izin berikut ke editor JSON.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:GetBucketVersioning"
            ],
            "Resource": "*"
        }
    ]
}
```

4. Pilih Tinjau kebijakan.
5. Di bawah Kebijakan peninjauan, untuk Nama kebijakan, masukkan**`s3-bucket-permissions`**.
6. Pilih Buat kebijakan.

Langkah 3: Jalankan eksekusi mesin status Standar

1. Pada halaman Gather-S3-Bucket-Info-Standard, pilih Mulai eksekusi.
2. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

    a. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

> **Note**
>
> Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

b. Pilih Mulai Eksekusi.

c. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

   Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Langkah 4: Jalankan eksekusi mesin status Express

1. Buat mesin status Express menggunakan definisi mesin negara yang disediakan di Langkah 1. Pastikan Anda juga menyertakan izin peran IAM yang diperlukan seperti yang dijelaskan pada Langkah 2.

   > **Tip**
   >
   > Untuk membedakan dari mesin Standar yang Anda buat sebelumnya, beri nama mesin status Express sebagai**`Gather-S3-Bucket-Info-Express`**.

2. Pada halaman Gather-S3-Bucket-Info-Standard, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

   a. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

> **Note**
>
> Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

b. Pilih Mulai Eksekusi.

c. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

   Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Alat developer

Sumber berikut berisi informasi tambahan tentang membangun alur kerja nirserver dan bekerja dengan mesin status:

- AWS CDK

- AWS Toolkit for VS Code

Topik di bawah ini berisi informasi yang mengajarkan Anda cara membuat, menguji, dan men-debug mesin status.

Topik

- Opsi Pengembangan
- AWS Step Functions dan AWS SAM
- Menggunakan Workflow Studio di Application Composer
- Membuat mesin status Lambda untuk Step Functions menggunakan AWS CloudFormation
- Membuat mesin Lambda negara untuk Step Functions digunakan AWS CDK
- Membuat API REST API Gateway dengan Mesin Status Ekspres Sinkron Menggunakan AWS CDK
- AWS Step Functions SDK Ilmu Data untuk Python
- Menyebarkan mesin status menggunakan Terraform

Opsi Pengembangan

Anda dapat mengimplementasikan mesin AWS Step Functions status Anda dengan beberapa cara, seperti menggunakan konsol, SDK, atau Step Functions versi lokal untuk pengujian dan pengembangan.

Topik

- Konsol Step Functions
- AWS SDK
- Alur kerja Standar dan Ekspres
- API Layanan HTTPS

- Lingkungan pengembangan
- Titik akhir
- AWS CLI
- Step Functions Local
- AWS Toolkit for Visual Studio Code
- AWS Serverless Application Model dan Step Functions
- Terraform dan Step Functions
- Dukungan format ketetapan

Konsol Step Functions

Anda dapat menentukan mesin status menggunakan konsol Step Functions. Anda dapat menulis mesin status kompleks di cloud tanpa menggunakan lingkungan pengembangan lokal dengan menggunakan AWS Lambda untuk menyediakan kode untuk tugas Anda. Setelah ditulis, Anda kemudian dapat menggunakan konsol Step Functions untuk menentukan mesin status Anda menggunakan Amazon States Language.

Tutorial Membuat Mesin Status Lambdatutorial menggunakan teknik ini untuk membuat mesin status sederhana, menjalankannya, dan melihat hasilnya.

Simulator aliran data

Anda dapat merancang, menerapkan dan men-debug alur kerja di konsol Step Functions. Anda juga dapat mengontrol aliran data dalam alur kerja Anda dengan menggunakan pemrosesan `JsonPath` input dan output. Gunakan simulator aliran data di konsol Step Functions untuk mempelajari bagaimana informasi mengalir dari satu status ke status lainnya, dan untuk memahami cara memfilter dan memanipulasi data. Alat ini mensimulasikan setiap bidang yang digunakan Step Functions untuk memproses data, seperti`InputPath`,`Parameters`, `ResultSelector``OutputPath`, dan`ResultPath`.

Untuk informasi selengkapnya, lihat Simulator aliran data.

AWS SDK

Step Functions didukung oleh AWS SDK untuk Java, .NET, Ruby, PHP, Python (Boto 3),, Go JavaScript, dan C ++. SDK ini menyediakan cara mudah untuk menggunakan tindakan Step Functions HTTPS API dalam beberapa bahasa pemrograman.

Anda dapat mengembangkan mesin status, kegiatan, atau pemicu mesin status menggunakan tindakan API yang diekspos pustaka SDK ini. Anda juga dapat mengakses operasi visibilitas dengan menggunakan pustaka ini untuk mengembangkan alat pemantauan dan pelaporan Step Functions Anda sendiri.

Untuk menggunakan Step Functions dengan AWS layanan lain, lihat dokumentasi referensi untuk AWS SDK dan Alat saat ini untuk Amazon Web Services.

> **Note**
>
> Step Functions hanya mendukung titik akhir HTTPS.

Alur kerja Standar dan Ekspres

Ketika Anda membuat mesin status baru, Anda harus memilih Type Standar atau Ekspres. Dalam kedua kasus tersebut, Anda menentukan mesin status menggunakan Amazon States Language. Eksekusi mesin status Anda akan berperilaku berbeda, tergantung pada Tipe yang Anda pilih. Jenis yang Anda pilih tidak dapat diubah setelah mesin status Anda dibuat.

Untuk informasi selengkapnya, lihat Logging menggunakanCloudWatchLog.

API Layanan HTTPS

Step Functions menyediakan operasi layanan yang dapat diakses melalui permintaan HTTPS. Anda dapat menggunakan operasi ini untuk berkomunikasi langsung dengan Step Functions, dan Anda dapat menggunakannya untuk mengembangkan pustaka Anda sendiri dalam bahasa apa pun yang dapat berkomunikasi dengan Step Functions melalui HTTPS.

Anda dapat mengembangkan mesin status, pekerja, atau pemicu mesin status dengan menggunakan tindakan API layanan. Anda juga dapat mengakses operasi visibilitas melalui tindakan API untuk mengembangkan alat pemantauan dan pelaporan Anda sendiri.

Untuk detail informasi API ini, lihat Referensi API AWS Step Functions.

Lingkungan pengembangan

Anda harus mengatur lingkungan pengembangan yang kompatibel dengan bahasa pemrograman yang Anda rencanakan untuk digunakan.

Misalnya, untuk mengembangkan Step Functions menggunakan Java, Anda harus menginstal lingkungan pengembangan Java, seperti AWS SDK for Java, pada setiap workstation pengembangan Anda. Jika Anda menggunakan IDE Eclipse untuk Java Developers, Anda juga harus untuk menginstal AWS Toolkit for Eclipse. Plugin Eclipse ini menambahkan fitur yang diperlukan untuk mengembangkan AWS.

Jika bahasa pemrograman Anda memerlukan lingkungan runtime, Anda harus mengatur lingkungan di setiap komputer tempat proses ini akan berjalan.

Titik akhir

Untuk mengurangi latensi dan menyimpan data di lokasi yang memenuhi persyaratan Anda, Step Functions menyediakan titik akhir di Wilayah yang berbeda AWS .

Setiap titik akhir dalam Step Functions sepenuhnya independen. Mesin status atau aktivitas hanya ada dalam Wilayah tempat ia dibuat. Setiap mesin status dan aktivitas yang Anda buat di suatu Wilayah tidak berbagi data atau atribut apa pun dengan yang dibuat di Wilayah lain. Misalnya, Anda dapat mendaftarkan mesin negara yang diberi nama `STATES-Flows-1` di dua Wilayah berbeda. Mesin `STATES-Flows-1` status di satu wilayah tidak akan berbagi data atau atribut dengan mesin `STATES-Flow-1` status di wilayah lain.

Untuk daftar titik akhir Step Functions, lihat AWS Step Functions Wilayah dan Titik Akhir di. Referensi Umum AWS

AWS CLI

Anda dapat mengakses banyak fitur Step Functions dari AWS Command Line Interface (AWS CLI). AWS CLI Ini adalah alternatif untuk menggunakan konsol Step Functions atau, dalam beberapa kasus, untuk pemrograman menggunakan tindakan Step Functions API. Misalnya, Anda dapat menggunakan AWS CLI untuk membuat mesin status dan kemudian mencantumkan mesin status yang ada.

Anda dapat menggunakan perintah Step Functions AWS CLI untuk memulai dan mengelola eksekusi, polling untuk aktivitas, merekam detak jantung tugas, dan banyak lagi. Untuk daftar lengkap perintah Step Functions, deskripsi argumen yang tersedia, dan contoh yang menunjukkan penggunaannya, lihat Referensi AWS CLI Perintah.

AWS CLI perintah mengikuti Amazon States Language dengan cermat, sehingga Anda dapat menggunakannya AWS CLI untuk mempelajari tentang tindakan Step Functions API. Anda juga

dapat menggunakan pengetahuan API yang ada untuk melakukan tindakan Step Functions dari baris perintah.

Step Functions Local

Untuk tujuan pengujian dan pengembangan, Anda dapat menginstal dan menjalankan Step Functions pada mesin lokal Anda. Dengan Step Functions Local, Anda dapat memulai eksekusi pada mesin apa pun.

Versi lokal Step Functions dapat memanggil AWS Lambda fungsi, baik di dalam AWS maupun saat berjalan secara lokal. Anda juga dapat mengoordinasikan AWS layanan lain yang didukung. Untuk informasi selengkapnya, lihat Menguji mesin negara secara lokal.

> **Note**
>
> Step Functions Local menggunakan akun dummy untuk bekerja.

AWS Toolkit for Visual Studio Code

Anda dapat menggunakan VS Code untuk berinteraksi dengan mesin status jarak jauh dan mengembangkan mesin status secara lokal. Anda dapat membuat atau memperbarui mesin status, membuat daftar mesin status yang ada, dan mengeksekusi atau mengunduh mesin status. Kode VS juga memungkinkan Anda membuat mesin status baru dari templat, melihat visualisasi mesin status Anda, dan menyediakan cuplikan kode, penyelesaian kode, serta validasi kode.

Untuk informasi selengkapnya, lihat Panduan AWS Toolkit for Visual Studio Code Pengguna

AWS Serverless Application Model dan Step Functions

Step Functions terintegrasi dengan AWS Serverless Application Model, yang memungkinkan Anda mengintegrasikan alur kerja dengan fungsi, API, dan peristiwa Lambda untuk membuat aplikasi tanpa server.

Anda juga dapat menggunakan AWS SAM CLI dalam hubungannya dengan AWS Toolkit for Visual Studio Code sebagai bagian dari pengalaman terintegrasi.

Lihat informasi yang lebih lengkap di AWS Step Functions dan AWS SAM.

Terraform dan Step Functions

Terraform by HashiCorp adalah kerangka kerja untuk membangun aplikasi menggunakan infrastruktur sebagai kode (IAc). Dengan Terraform, Anda dapat membuat mesin status dan menggunakan fitur, seperti melihat pratinjau penerapan infrastruktur dan membuat templat yang dapat digunakan kembali. Template Terraform membantu Anda memelihara dan menggunakan kembali kode dengan memecahnya menjadi potongan-potongan yang lebih kecil.

Untuk informasi selengkapnya, lihat Menyebarkan mesin status menggunakan Terraform.

Dukungan format ketetapan

Step Functions menawarkan berbagai alat yang memungkinkan Anda memberikan definisi mesin status Anda dalam format yang berbeda. Definisi Amazon States Language (ASL) yang menentukan detail mesin status Anda dapat diberikan baik sebagai string, atau sebagai objek serial menggunakan JSON atau YAMAL.

> **Note**
>
> YAML memungkinkan komentar baris tunggal. Komentar YAML yang disediakan di bagian ketetapan mesin status templat tidak akan dibawa ke ketetapan sumber daya yang dibuat. Sebaliknya, Anda dapat menggunakan properti `Comment` dalam ketetapan mesin status. Untuk informasi selengkapnya, lihat halaman Struktur mesin status.

Tabel berikut menunjukkan alat yang mendukung ketetapan berbasis ASL.

Format ketetapan dukungan oleh alat

	JSON	YAML	Stringified Amazon States Language		
Step Functions Console	✓				

	JSON	YAML	Stringified Amazon States Language		
API Layanan HTTPS			✓		
AWS CLI			✓		
Step Functions Lokal			✓		
AWS Toolkit for Visual Studio Code	✓	✓			
AWS SAM	✓	✓			
AWS CloudFormation	✓	✓	✓		

> **Note**
>
> AWS CloudFormation dan AWS SAM juga memungkinkan Anda untuk mengunggah definisi mesin status Anda ke Amazon S3 dalam format JSON atau YAMAL, dan untuk memberikan lokasi Amazon S3 definisi dalam template. Hal ini dapat meningkatkan keterbacaan templat Anda ketika ketetapan mesin status Anda kompleks. Untuk informasi lebih lanjut, lihat halaman AWS::StepFunctions::StateMachine S3Location.

Contoh AWS CloudFormation templat berikut menunjukkan bagaimana Anda dapat memberikan definisi mesin status yang sama menggunakan format input yang berbeda.

JSON with Definition

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "AWS Step Functions sample template.",
  "Resources": {
    "MyStateMachine": {
      "Type": "AWS::StepFunctions::StateMachine",
      "Properties": {
        "RoleArn": {
          "Fn::GetAtt": [ "StateMachineRole", "Arn" ]
        },
        "TracingConfiguration": {
          "Enabled": true
        },
        "Definition": {
          "StartAt": "HelloWorld",
          "States": {
            "HelloWorld": {
              "Type": "Pass",
              "End": true
            }
          }
        }
      }
    },
    "StateMachineRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Action": [
                "sts:AssumeRole"
              ],
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  "states.amazonaws.com"
                ]
              }
            }
          ]
```

```
        },
        "ManagedPolicyArns": [],
        "Policies": [
          {
            "PolicyName": "StateMachineRolePolicy",
            "PolicyDocument": {
              "Statement": [
                {
                  "Action": [
                    "lambda:InvokeFunction"
                  ],
                  "Resource": "*",
                  "Effect": "Allow"
                }
              ]
            }
          }
        ]
      }
    }
  },
  "Outputs": {
    "StateMachineArn": {
      "Value": {
        "Ref": "MyStateMachine"
      }
    }
  }
}
```

JSON with DefinitionString

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "AWS Step Functions sample template.",
  "Resources": {
    "MyStateMachine": {
      "Type": "AWS::StepFunctions::StateMachine",
      "Properties": {
        "RoleArn": {
          "Fn::GetAtt": [ "StateMachineRole", "Arn" ]
        },
        "TracingConfiguration": {
```

```
          "Enabled": true
        },
        "DefinitionString": "{\n  \"StartAt\": \"HelloWorld\",\n  \"States\": {\n
\"HelloWorld\": {\n      \"Type\": \"Pass\",\n      \"End\": true\n    }\n  }\n}"
      }
    },
    "StateMachineRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Action": [
                "sts:AssumeRole"
              ],
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  "states.amazonaws.com"
                ]
              }
            }
          ]
        },
        "ManagedPolicyArns": [],
        "Policies": [
          {
            "PolicyName": "StateMachineRolePolicy",
            "PolicyDocument": {
              "Statement": [
                {
                  "Action": [
                    "lambda:InvokeFunction"
                  ],
                  "Resource": "*",
                  "Effect": "Allow"
                }
              ]
            }
          }
        ]
      }
    }
```

```
  },
  "Outputs": {
    "StateMachineArn": {
      "Value": {
        "Ref": "MyStateMachine"
      }
    }
  }
}
```

YAML with Definition

```
AWSTemplateFormatVersion: 2010-09-09
Description: AWS Step Functions sample template.
Resources:
  MyStateMachine:
    Type: 'AWS::StepFunctions::StateMachine'
    Properties:
      RoleArn: !GetAtt
        - StateMachineRole
        - Arn
      TracingConfiguration:
        Enabled: true
      Definition:
        # This is a YAML comment. This will not be preserved in the state machine
 resource's definition.
        Comment: This is an ASL comment. This will be preserved in the state machine
 resource's definition.
        StartAt: HelloWorld
        States:
          HelloWorld:
            Type: Pass
            End: true
  StateMachineRole:
    Type: 'AWS::IAM::Role'
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Action:
              - 'sts:AssumeRole'
            Effect: Allow
            Principal:
```

```
              Service:
                - states.amazonaws.com
      ManagedPolicyArns: []
      Policies:
        - PolicyName: StateMachineRolePolicy
          PolicyDocument:
            Statement:
              - Action:
                  - 'lambda:InvokeFunction'
                Resource: "*"
                Effect: Allow

Outputs:
  StateMachineArn:
    Value:
      Ref: MyStateMachine
```

YAML with DefinitionString

```
AWSTemplateFormatVersion: 2010-09-09
Description: AWS Step Functions sample template.
Resources:
  MyStateMachine:
    Type: 'AWS::StepFunctions::StateMachine'
    Properties:
      RoleArn: !GetAtt
        - StateMachineRole
        - Arn
      TracingConfiguration:
        Enabled: true
      DefinitionString: |
        {
            "StartAt": "HelloWorld",
            "States": {
                "HelloWorld": {
                    "Type": "Pass",
                    "End": true
                }
            }
        }
  StateMachineRole:
    Type: 'AWS::IAM::Role'
    Properties:
```

```
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Action:
              - 'sts:AssumeRole'
            Effect: Allow
            Principal:
              Service:
                - states.amazonaws.com
      ManagedPolicyArns: []
      Policies:
        - PolicyName: StateMachineRolePolicy
          PolicyDocument:
            Statement:
              - Action:
                  - 'lambda:InvokeFunction'
                Resource: "*"
                Effect: Allow

Outputs:
  StateMachineArn:
    Value:
      Ref: MyStateMachine
```

AWS Step Functions dan AWS SAM

Anda dapat menggunakan AWS SAM CLI bersama dengan AWS Toolkit for Visual Studio Code sebagai bagian dari pengalaman terintegrasi untuk membuat mesin negara secara lokal. Anda dapat membuat aplikasi nirserver dengan AWS SAM, kemudian membangun mesin status Anda di VS Code IDE. Kemudian Anda dapat memvalidasi, mengemas, dan menyebarkan sumber daya Anda. Secara opsional, Anda juga dapat mempublikasikan ke. AWS Serverless Application Repository

> **Tip**
>
> Untuk menerapkan contoh aplikasi tanpa server yang memulai alur kerja Step Functions menggunakan aplikasi Anda Akun AWS, lihat Modul 11 - AWS SAM Menerapkan dengan Workshop. AWS SAM AWS Step Functions

Topik

- Mengapa menggunakan Step Functions dengan AWS SAM?
- Integrasi Step Functions dengan spesifikasi AWS SAM
- Integrasi Step Functions dengan SAM CLI
- DefinitionSubstitutions dalam AWS SAM template
- Langkah selanjutnya

Mengapa menggunakan Step Functions dengan AWS SAM?

Bila Anda menggunakan Step Functions dengan AWS SAM Anda dapat:

- Mulai menggunakan template AWS SAM sampel.
- Membangun mesin status Anda ke dalam aplikasi nirserver.
- Gunakan substitusi variabel untuk mengganti ARN ke mesin status Anda pada saat penerapan.

  AWS CloudFormationmendukung `DefinitionSubstitutions`yang memungkinkan Anda menambahkan referensi dinamis dalam definisi alur kerja Anda ke nilai yang Anda berikan di CloudFormation template Anda. Anda dapat menambahkan referensi dinamis dengan menambahkan substitusi ke definisi alur kerja Anda menggunakan notasi. `${dollar_sign_brace}` Anda juga perlu mendefinisikan referensi dinamis ini di `DefinitionSubstitutions` properti untuk StateMachine sumber daya di CloudFormation template Anda. Substitusi ini diganti dengan nilai aktual selama proses pembuatan CloudFormation tumpukan. Untuk informasi selengkapnya, lihat DefinitionSubstitutions dalam AWS SAM template.
- Tentukan peran mesin status Anda menggunakan templat AWS SAM kebijakan.
- Memulai eksekusi mesin status dengan API Gateway, EventBridge peristiwa, atau pada jadwal dalam template Anda AWS SAM .

Integrasi Step Functions dengan spesifikasi AWS SAM

Anda dapat menggunakan Templat AWS SAM Kebijakan untuk menambahkan izin ke mesin status Anda. Dengan izin ini, Anda dapat mengatur fungsi Lambda dan sumber daya AWS lainnya untuk membentuk alur kerja yang kompleks dan kuat.

Integrasi Step Functions dengan SAM CLI

Step Functions terintegrasi dengan AWS SAM CLI. Gunakan ini untuk mengembangkan mesin status Anda ke dalam aplikasi nirserver dengan cepat.

Coba Membuat mesin kondisi Step FunctionsAWS SAM tutorial untuk mempelajari cara menggunakan AWS SAM untuk membuat mesin negara.

Fungsi AWS SAM CLI yang didukung meliputi:

Perintah CLI	Deskripsi
init sam	Menginisialisasi Aplikasi Tanpa Server dengan template. AWS SAM Dapat digunakan dengan templat SAM untuk Step Functions.
validasi sam	Memvalidasi AWS SAM template.
paket sam	Paket AWS SAM aplikasi. Ini membuat file ZIP kode dan dependensi Anda, dan kemudian mengunggahnya ke Amazon S3. Kemudian mengembalikan salinan templat AWS SAM Anda, menggantikan referensi ke artefak lokal dengan lokasi Amazon S3 tempat perintah mengunggah artefak.
deploy sam	Menyebarkan AWS SAM aplikasi.
terbitan sam	Publikasikan AWS SAM aplikasi ke AWS Serverless Application Repository. Perintah ini mengambil AWS SAM template yang dikemas dan menerbitkan aplikasi ke wilayah yang ditentukan.

> **Note**
>
> Saat menggunakan AWS SAM lokal, Anda dapat meniru Lambda dan API Gateway secara lokal. Namun, Anda tidak dapat meniru Step Functions secara lokal menggunakan. AWS SAM

DefinitionSubstitutions dalam AWS SAM template

Anda dapat menentukan mesin negara menggunakan CloudFormation templat denganAWS SAM. Dengan menggunakan AWS SAM, Anda dapat mendefinisikan inline mesin status di templat atau di file terpisah. AWS SAMTemplate berikut mencakup mesin negara yang mensimulasikan alur kerja perdagangan saham. Mesin negara ini menggunakan tiga Lambda fungsi untuk memeriksa harga saham dan menentukan apakah akan membeli atau menjual saham. Transaksi ini kemudian dicatat dalam Amazon DynamoDB tabel. ARN untuk Lambda fungsi dan DynamoDB tabel dalam template berikut ditentukan menggunakan `DefinitionSubstitutions`.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: |
  step-functions-stock-trader
  Sample SAM Template for step-functions-stock-trader
Resources:
  StockTradingStateMachine:
    Type: AWS::Serverless::StateMachine
    Properties:
      DefinitionSubstitutions:
        StockCheckerFunctionArn: !GetAtt StockCheckerFunction.Arn
        StockSellerFunctionArn: !GetAtt StockSellerFunction.Arn
        StockBuyerFunctionArn: !GetAtt StockBuyerFunction.Arn
        DDBPutItem: !Sub arn:${AWS::Partition}:states:::dynamodb:putItem
        DDBTable: !Ref TransactionTable
      Policies:
        - DynamoDBWritePolicy:
            TableName: !Ref TransactionTable
        - LambdaInvokePolicy:
            FunctionName: !Ref StockCheckerFunction
        - LambdaInvokePolicy:
            FunctionName: !Ref StockBuyerFunction
        - LambdaInvokePolicy:
            FunctionName: !Ref StockSellerFunction
      DefinitionUri: statemachine/stock_trader.asl.json
  StockCheckerFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: functions/stock-checker/
      Handler: app.lambdaHandler
      Runtime: nodejs18.x
      Architectures:
```

```
        - x86_64
  StockSellerFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: functions/stock-seller/
      Handler: app.lambdaHandler
      Runtime: nodejs18.x
      Architectures:
        - x86_64
  StockBuyerFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: functions/stock-buyer/
      Handler: app.lambdaHandler
      Runtime: nodejs18.x
      Architectures:
        - x86_64
  TransactionTable:
    Type: AWS::DynamoDB::Table
    Properties:
      AttributeDefinitions:
        - AttributeName: id
          AttributeType: S
```

Kode berikut adalah definisi mesin negara dalam file `stock_trader.asl.json` yang digunakan dalam Membuat mesin kondisi Step FunctionsAWS SAM tutorial. Definisi mesin negara ini berisi beberapa `DefinitionSubstitutions` dilambangkan dengan notasi. `${dollar_sign_brace}` Misalnya, alih-alih menentukan ARN Lambda fungsi statis untuk tugas `Check Stock Value` tersebut, `${StockCheckerFunctionArn}` substitusi digunakan. Substitusi ini didefinisikan dalam DefinitionSubstitutions properti template. `DefinitionSubstitutions`adalah peta pasangan kunci-nilai untuk sumber daya mesin negara. Dalam`DefinitionSubstitutions`, $ {StockCheckerFunctionArn} memetakan ke ARN `StockCheckerFunction` sumber daya menggunakan fungsi CloudFormation intrinsik. `!GetAtt` Saat Anda menyebarkan AWS SAM template, `DefinitionSubstitutions` dalam template diganti dengan nilai sebenarnya.

```
{
    "Comment": "A state machine that does mock stock trading.",
    "StartAt": "Check Stock Value",
    "States": {
        "Check Stock Value": {
            "Type": "Task",
            "Resource": "arn:aws:states:::lambda:invoke",
```

```
            "OutputPath": "$.Payload",
            "Parameters": {
                "Payload.$": "$",
                "FunctionName": "${StockCheckerFunctionArn}"
            },
            "Next": "Buy or Sell?"
        },
        "Buy or Sell?": {
            "Type": "Choice",
            "Choices": [
                {
                    "Variable": "$.stock_price",
                    "NumericLessThanEquals": 50,
                    "Next": "Buy Stock"
                }
            ],
            "Default": "Sell Stock"
        },
        "Buy Stock": {
            "Type": "Task",
            "Resource": "arn:aws:states:::lambda:invoke",
            "OutputPath": "$.Payload",
            "Parameters": {
                "Payload.$": "$",
                "FunctionName": "${StockBuyerFunctionArn}"
            },
            "Retry": [
                {
                    "ErrorEquals": [
                        "Lambda.ServiceException",
                        "Lambda.AWSLambdaException",
                        "Lambda.SdkClientException",
                        "Lambda.TooManyRequestsException"
                    ],
                    "IntervalSeconds": 1,
                    "MaxAttempts": 3,
                    "BackoffRate": 2
                }
            ],
            "Next": "Record Transaction"
        },
        "Sell Stock": {
            "Type": "Task",
            "Resource": "arn:aws:states:::lambda:invoke",
```

```
            "OutputPath": "$.Payload",
            "Parameters": {
                "Payload.$": "$",
                "FunctionName": "${StockSellerFunctionArn}"
            },
            "Next": "Record Transaction"
        },
        "Record Transaction": {
            "Type": "Task",
            "Resource": "arn:aws:states:::dynamodb:putItem",
            "Parameters": {
                "TableName": "${DDBTable}",
                "Item": {
                    "Id": {
                        "S.$": "$.id"
                    },
                    "Type": {
                        "S.$": "$.type"
                    },
                    "Price": {
                        "N.$": "$.price"
                    },
                    "Quantity": {
                        "N.$": "$.qty"
                    },
                    "Timestamp": {
                        "S.$": "$.timestamp"
                    }
                }
            },
            "End": true
        }
    }
}
```

Langkah selanjutnya

Anda dapat mempelajari selengkapnya tentang penggunaan Step Functions AWS SAM dengan sumber daya berikut:

- Lengkapi Membuat mesin kondisi Step FunctionsAWS SAM tutorial untuk membuat mesin negara dengan AWS SAM.

- Tentukan sumber AWS::Serverless::StateMachinedaya.
- Temukan Templat kebijakan AWS SAM untuk digunakan.
- Gunakan AWS Toolkit for Visual Studio Code dengan Step Functions.
- Tinjau referensi AWS SAM CLI untuk mempelajari lebih lanjut tentang fitur yang tersedia di. AWS SAM

Anda juga dapat mendesain dan membangun alur kerja Anda di infrastruktur sebagai kode (IAc) menggunakan pembuat visual, seperti Workflow Studio in. Application Composer Untuk informasi selengkapnya, lihat Menggunakan Workflow Studio di Application Composer.

Menggunakan Workflow Studio di Application Composer

Komposer Aplikasi AWSadalah pembangun visual yang membantu Anda mengembangkan AWS SAM dan membuat AWS CloudFormation templat menggunakan antarmuka grafis sederhana. DenganApplication Composer, Anda merancang arsitektur aplikasi dengan menyeret, mengelompokkan, dan menghubungkan Layanan AWS dalam kanvas visual. Application Composerkemudian membuat infrastruktur sebagai kode (IAc) template dari desain Anda yang dapat Anda gunakan untuk menyebarkan aplikasi Anda dengan AWS SAM Command Line Interface (AWS SAMCLI) atau. CloudFormation Untuk mempelajari lebih lanjut tentangApplication Composer, lihat Apa itu Application Composer.

Workflow Studio tersedia Application Composer untuk membantu Anda merancang dan membangun alur kerja Anda. Workflow Studio in Application Composer menyediakan lingkungan IAc visual yang memudahkan Anda untuk menggabungkan alur kerja dalam aplikasi tanpa server yang dibangun menggunakan alat IAc, seperti template. CloudFormation Saat Anda menggunakan Workflow Studio inApplication Composer, ini menghubungkan langkah alur kerja individual ke AWS sumber daya dan menghasilkan konfigurasi sumber daya dalam templat. AWS SAM Ini juga menambahkan IAM izin yang diperlukan agar alur kerja Anda berjalan. Menggunakan Workflow Studio inApplication Composer, Anda dapat membuat prototipe aplikasi Anda dan mengubahnya menjadi aplikasi siap produksi.

Saat Anda menggunakan Workflow Studio diApplication Composer, Anda dapat beralih bolak-balik antara Application Composer kanvas dan Workflow Studio.

Topik

- Menggunakan Workflow Studio Application Composer untuk membangun alur kerja tanpa server

- Referensi sumber daya secara dinamis menggunakan substitusi CloudFormation definisi di Workflow Studio
- Connect tugas integrasi layanan ke kartu komponen yang disempurnakan
- Impor proyek yang ada dan sinkronkan secara lokal
- Fitur Workflow Studio yang tidak tersedia di Komposer Aplikasi AWS

Menggunakan Workflow Studio Application Composer untuk membangun alur kerja tanpa server

1. Buka konsol Application Composer dan pilih Create project untuk membuat proyek.
2. Di bidang pencarian di palet Sumber Daya, masukkan**`state machine`**.
3. Seret sumber daya mesin Step Functions State ke kanvas.
4. Pilih Edit di Workflow Studio untuk mengedit sumber daya mesin status Anda.

   Animasi berikut menunjukkan bagaimana Anda dapat beralih ke Workflow Studio untuk mengedit definisi mesin status Anda.
   Animasi yang menggambarkan bagaimana Anda dapat menggunakan Workflow Studio di. Application Composer

   Integrasi dengan Workflow Studio untuk mengedit sumber daya state machine yang dibuat hanya Application Composer tersedia untuk `AWS::Serverless::StateMachine`sumber daya. Integrasi ini tidak tersedia untuk template yang menggunakan `AWS::StepFunctions::StateMachine`sumber daya.

Referensi sumber daya secara dinamis menggunakan substitusi CloudFormation definisi di Workflow Studio

Di Workflow Studio, Anda dapat menggunakan substitusi CloudFormation definisi dalam definisi alur kerja untuk mereferensikan sumber daya secara dinamis yang telah Anda tetapkan dalam template IAc Anda. Anda dapat menambahkan substitusi placeholder ke definisi alur kerja Anda menggunakan `${dollar_sign_brace}` notasi dan diganti dengan nilai aktual selama proses pembuatan tumpukan. CloudFormation Untuk informasi lebih lanjut tentang substitusi definisi, lihat. DefinitionSubstitutions dalam AWS SAM template

Animasi berikut menunjukkan bagaimana Anda dapat menambahkan substitusi placeholder untuk sumber daya dalam definisi mesin status Anda.
Animasi yang menggambarkan cara mereferensikan sumber daya secara dinamis, seperti AWS Lambda fungsi, substitusi definisi saat Anda menggunakan Workflow Studio. Application Composer

Connect tugas integrasi layanan ke kartu komponen yang disempurnakan

Anda dapat menghubungkan tugas yang memanggil integrasi layanan yang dioptimalkan ke kartu komponen yang disempurnakan di Application Composer kanvas. Melakukan hal ini secara otomatis memetakan setiap substitusi placeholder yang ditentukan oleh `${dollar_sign_brace}` notasi dalam definisi alur kerja Anda dan properti untuk sumber daya Anda. `DefinitionSubstitution` `StateMachine` Ini juga menambahkan AWS SAM kebijakan yang sesuai untuk mesin negara.

Jika Anda memetakan tugas integrasi layanan yang dioptimalkan dengan kartu komponen standar, jalur koneksi tidak muncul di Application Composer kanvas.

Animasi berikut menunjukkan bagaimana Anda dapat menghubungkan tugas yang dioptimalkan ke kartu komponen yang disempurnakan dan melihat perubahan di Change Inspector.
Animasi yang menggambarkan cara menghubungkan tugas yang memanggil integrasi layanan yang dioptimalkan ke kartu komponen yang disempurnakan saat Anda menggunakan Workflow Studio in. Application Composer

Anda tidak dapat menghubungkan integrasi AWS SDK dalam status Tugas dengan kartu komponen yang disempurnakan atau integrasi layanan yang dioptimalkan dengan kartu komponen standar. Untuk tugas ini, Anda dapat memetakan substitusi di panel Resource properties di Application Composer canvas, dan menambahkan kebijakan dalam template. AWS SAM

> **Tip**
>
> Atau, Anda juga dapat memetakan substitusi placeholder untuk mesin status Anda di bawah Substitusi Definisi di panel properti Resource. Ketika Anda melakukan ini, Anda harus menambahkan izin yang diperlukan untuk panggilan status Tugas Layanan AWS Anda dalam peran eksekusi mesin status. Untuk informasi tentang izin yang mungkin diperlukan peran eksekusi Anda, lihatPeran eksekusi di Workflow Studio.

Animasi berikut menunjukkan bagaimana Anda dapat memperbarui pemetaan substitusi placeholder secara manual di panel properti Resource.

Animasi yang mengilustrasikan cara memperbarui pemetaan substitusi placeholder secara manual di panel properti Resource saat Anda menggunakan Workflow Studio in. Application Composer

Impor proyek yang ada dan sinkronkan secara lokal

Anda dapat membuka yang ada CloudFormation dan AWS SAM proyek Application Composer untuk memvisualisasikannya untuk pemahaman yang lebih baik dan memodifikasi desain mereka. Menggunakan Application Composer fitur sinkronisasi lokal, Anda dapat secara otomatis menyinkronkan dan menyimpan file template dan kode Anda ke mesin build lokal Anda. Menggunakan mode sinkronisasi lokal dapat melengkapi alur pengembangan yang ada. Pastikan browser Anda mendukung File System Access API, yang memungkinkan aplikasi web membaca, menulis, dan menyimpan file di sistem file lokal Anda. Sebaiknya gunakan Google Chrome atau Microsoft Edge.

Fitur Workflow Studio yang tidak tersedia di Komposer Aplikasi AWS

Saat Anda menggunakan Workflow Studio diApplication Composer, beberapa fitur Workflow Studio tidak tersedia. Selain itu, bagian Parameter API yang tersedia di Inspector panel mendukung substitusi CloudFormation definisi. Anda dapat menambahkan substitusi dalam Mode kode menggunakan notasi. `${dollar_sign_brace}` Untuk informasi lebih lanjut tentang notasi ini, lihatDefinitionSubstitutions dalam AWS SAM template.

Daftar berikut menjelaskan fitur Workflow Studio yang tidak tersedia saat Anda menggunakan Workflow Studio di: Application Composer

- Template pemula - Template pemula adalah proyek ready-to-run sampel yang secara otomatis membuat proptotipe dan definisi alur kerja. Template ini menyebarkan semua AWS sumber daya terkait yang dibutuhkan proyek AndaAkun AWS.
- Mode Config — Mode ini memungkinkan Anda mengelola konfigurasi mesin status Anda. Anda dapat memperbarui konfigurasi mesin status Anda di templat IAC Anda atau menggunakan panel properti Sumber Daya di Application Composer kanvas. Untuk informasi tentang memperbarui konfigurasi di panel Properti sumber daya, lihatConnect tugas integrasi layanan ke kartu komponen yang disempurnakan.
- API TestState
- Opsi untuk mengimpor atau mengekspor definisi alur kerja dari tombol dropdown Tindakan di Workflow Studio. Sebagai gantinya, dari Application Composer menu, pilih Open > Project folder. Pastikan Anda telah mengaktifkan mode sinkronisasi lokal untuk secara otomatis menyimpan perubahan Anda di Application Composer kanvas langsung ke mesin lokal Anda.

- Jalankan tombol. Saat Anda menggunakan Workflow Studio diApplication Composer, Application Composer buat kode IAc untuk alur kerja Anda. Oleh karena itu, Anda harus terlebih dahulu menyebarkan template. Kemudian, jalankan alur kerja di konsol atau melalui file. AWS Command Line Interface (AWS CLI)

Membuat mesin status Lambda untuk Step Functions menggunakan AWS CloudFormation

Tutorial ini menunjukkan cara membuat AWS Lambda fungsi dasar menggunakan AWS CloudFormation. Anda akan menggunakan AWS CloudFormation konsol dan template YAMAL untuk membuat tumpukan (peran IAM, fungsi Lambda, dan mesin status). Kemudian, Anda akan menggunakan AWS Step Functions konsol untuk memulai eksekusi mesin status.

Untuk informasi selengkapnya, lihat Bekerja dengan CloudFormation Template dan `AWS::StepFunctions::StateMachine` sumber daya di Panduan AWS CloudFormation Pengguna.

Topik

- Langkah 1: Siapkan AWS CloudFormation template Anda
- Langkah 2: Gunakan AWS CloudFormation template untuk membuat Lambda State Machine
- Langkah 3: Mulai eksekusi Mesin Status

Langkah 1: Siapkan AWS CloudFormation template Anda

Sebelum Anda menggunakan Templat contoh, Anda harus memahami cara mendeklarasikan bagian-bagian berbeda dari sebuah templat AWS CloudFormation .

Topik

- Untuk membuat IAM role untuk Lambda
- Untuk membuat fungsi Lambda
- Untuk membuat IAM role untuk eksekusi mesin status
- Untuk membuat mesin status Lambda

Untuk membuat IAM role untuk Lambda

Tentukan kebijakan kepercayaan yang terkait dengan IAM role untuk fungsi Lambda. Contoh berikut mendefinisikan kebijakan kepercayaan menggunakan YAMAL atau JSON.

YAML

```
LambdaExecutionRole:
  Type: "AWS::IAM::Role"
  Properties:
    AssumeRolePolicyDocument:
      Version: "2012-10-17"
      Statement:
        - Effect: Allow
          Principal:
            Service: lambda.amazonaws.com
          Action: "sts:AssumeRole"
```

JSON

```
        "LambdaExecutionRole": {
          "Type": "AWS::IAM::Role",
          "Properties": {
              "AssumeRolePolicyDocument": {
                  "Version": "2012-10-17",
                  "Statement": [
                      {
                          "Effect": "Allow",
                          "Principal": {
                              "Service": "lambda.amazonaws.com"
                          },
                          "Action": "sts:AssumeRole"
                      }
                  ]
              }
          }
```

Untuk membuat fungsi Lambda

Tentukan properti berikut untuk fungsi Lambda yang akan mencetak pesan. `Hello World`

> **Important**
>
> Pastikan fungsi Lambda Anda berada di bawah AWS akun dan AWS Wilayah yang sama dengan mesin negara bagian Anda.

YAML

```
MyLambdaFunction:
  Type: "AWS::Lambda::Function"
  Properties:
    Handler: "index.handler"
    Role: !GetAtt [ LambdaExecutionRole, Arn ]
    Code:
      ZipFile: |
        exports.handler = (event, context, callback) => {
            callback(null, "Hello World!");
        };
    Runtime: "nodejs12.x"
    Timeout: "25"
```

JSON

```
        "MyLambdaFunction": {
            "Type": "AWS::Lambda::Function",
            "Properties": {
                "Handler": "index.handler",
                "Role": {
                    "Fn::GetAtt": [
                        "LambdaExecutionRole",
                        "Arn"
                    ]
                },
                "Code": {
                    "ZipFile": "exports.handler = (event, context, callback) => {\n
   callback(null, \"Hello World!\");\n};\n"
                },
                "Runtime": "nodejs12.x",
                "Timeout": "25"
            }
        },
```

Untuk membuat IAM role untuk eksekusi mesin status

Tentukan kebijakan kepercayaan yang terkait dengan IAM role untuk eksekusi mesin status.

YAML

```
StatesExecutionRole:
  Type: "AWS::IAM::Role"
  Properties:
    AssumeRolePolicyDocument:
      Version: "2012-10-17"
      Statement:
        - Effect: "Allow"
          Principal:
            Service:
              - !Sub states.${AWS::Region}.amazonaws.com
          Action: "sts:AssumeRole"
    Path: "/"
    Policies:
      - PolicyName: StatesExecutionPolicy
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
            - Effect: Allow
              Action:
                - "lambda:InvokeFunction"
              Resource: "*"
```

JSON

```
        "StatesExecutionRole": {
            "Type": "AWS::IAM::Role",
            "Properties": {
                "AssumeRolePolicyDocument": {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {
                                "Service": [
                                    {
                                        "Fn::Sub": "states.
${AWS::Region}.amazonaws.com"
```

```
                                }
                            ]
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            },
            "Path": "/",
            "Policies": [
                {
                    "PolicyName": "StatesExecutionPolicy",
                    "PolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Action": [
                                    "lambda:InvokeFunction"
                                ],
                                "Resource": "*"
                            }
                        ]
                    }
                }
            ]
        }
    },
```

Untuk membuat mesin status Lambda

Tentukan mesin status Lambda.

YAML

```
MyStateMachine:
  Type: "AWS::StepFunctions::StateMachine"
  Properties:
    DefinitionString:
      !Sub
        - |-
          {
            "Comment": "A Hello World example using an AWS Lambda function",
```

```
            "StartAt": "HelloWorld",
            "States": {
              "HelloWorld": {
                "Type": "Task",
                "Resource": "${lambdaArn}",
                "End": true
              }
            }
          }
        - {lambdaArn: !GetAtt [ MyLambdaFunction, Arn ]}
    RoleArn: !GetAtt [ StatesExecutionRole, Arn ]
```

JSON

```
        "MyStateMachine": {
            "Type": "AWS::StepFunctions::StateMachine",
            "Properties": {
                "DefinitionString": {
                    "Fn::Sub": [
                        "{\n  \"Comment\": \"A Hello World example using an
 AWS Lambda function\",\n  \"StartAt\": \"HelloWorld\",\n  \"States\": {\n
 \"HelloWorld\": {\n      \"Type\": \"Task\",\n      \"Resource\": \"${lambdaArn}\",
\n      \"End\": true\n    }\n  }\n}",
                        {
                            "lambdaArn": {
                                "Fn::GetAtt": [
                                    "MyLambdaFunction",
                                    "Arn"
                                ]
                            }
                        }
                    ]
                },
                "RoleArn": {
                    "Fn::GetAtt": [
                        "StatesExecutionRole",
                        "Arn"
                    ]
                }
            }
        }
```

Langkah 2: Gunakan AWS CloudFormation template untuk membuat Lambda State Machine

Setelah Anda memahami komponen AWS CloudFormation template, Anda dapat menyatukannya dan menggunakan template untuk membuat AWS CloudFormation tumpukan.

Untuk membuat mesin status Lambda

1. Salin data contoh berikut ke file bernama `MyStateMachine.yaml` untuk contoh YAML, atau `MyStateMachine.json` untuk JSON.

   YAML

```
AWSTemplateFormatVersion: "2010-09-09"
Description: "An example template with an IAM role for a Lambda state machine."
Resources:
  LambdaExecutionRole:
    Type: "AWS::IAM::Role"
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Principal:
              Service: lambda.amazonaws.com
            Action: "sts:AssumeRole"

  MyLambdaFunction:
    Type: "AWS::Lambda::Function"
    Properties:
      Handler: "index.handler"
      Role: !GetAtt [ LambdaExecutionRole, Arn ]
      Code:
        ZipFile: |
          exports.handler = (event, context, callback) => {
              callback(null, "Hello World!");
          };
      Runtime: "nodejs12.x"
      Timeout: "25"

  StatesExecutionRole:
    Type: "AWS::IAM::Role"
```

```
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: "Allow"
            Principal:
              Service:
                - !Sub states.${AWS::Region}.amazonaws.com
            Action: "sts:AssumeRole"
      Path: "/"
      Policies:
        - PolicyName: StatesExecutionPolicy
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
              - Effect: Allow
                Action:
                  - "lambda:InvokeFunction"
                Resource: "*"

  MyStateMachine:
    Type: "AWS::StepFunctions::StateMachine"
    Properties:
      DefinitionString:
        !Sub
          - |-
            {
              "Comment": "A Hello World example using an AWS Lambda function",
              "StartAt": "HelloWorld",
              "States": {
                "HelloWorld": {
                  "Type": "Task",
                  "Resource": "${lambdaArn}",
                  "End": true
                }
              }
            }
          - {lambdaArn: !GetAtt [ MyLambdaFunction, Arn ]}
      RoleArn: !GetAtt [ StatesExecutionRole, Arn ]
```

JSON

```
{
```

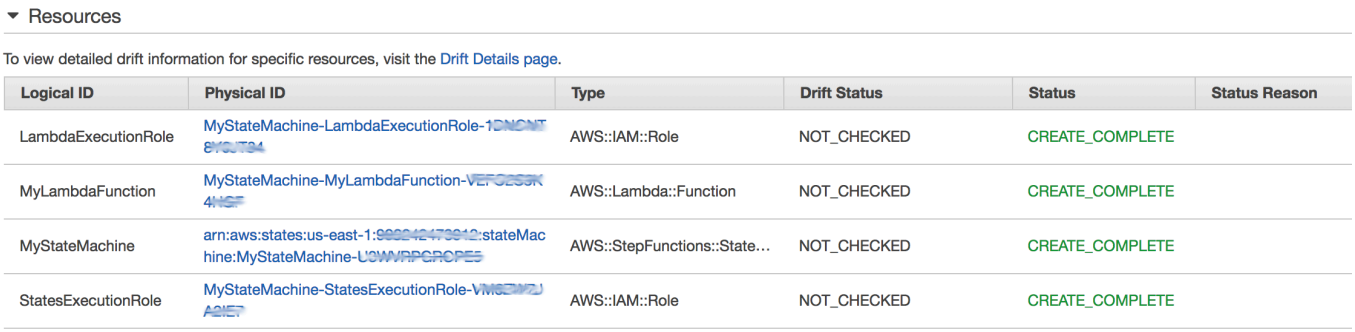
```
   "AWSTemplateFormatVersion": "2010-09-09",
   "Description": "An example template with an IAM role for a Lambda state
machine.",
   "Resources": {
       "LambdaExecutionRole": {
           "Type": "AWS::IAM::Role",
           "Properties": {
               "AssumeRolePolicyDocument": {
                   "Version": "2012-10-17",
                   "Statement": [
                       {
                           "Effect": "Allow",
                           "Principal": {
                               "Service": "lambda.amazonaws.com"
                           },
                           "Action": "sts:AssumeRole"
                       }
                   ]
               }
           }
       },
       "MyLambdaFunction": {
           "Type": "AWS::Lambda::Function",
           "Properties": {
               "Handler": "index.handler",
               "Role": {
                   "Fn::GetAtt": [
                       "LambdaExecutionRole",
                       "Arn"
                   ]
               },
               "Code": {
                   "ZipFile": "exports.handler = (event, context, callback) =>
{\n    callback(null, \"Hello World!\");\n};\n"
               },
               "Runtime": "nodejs12.x",
               "Timeout": "25"
           }
       },
       "StatesExecutionRole": {
           "Type": "AWS::IAM::Role",
           "Properties": {
               "AssumeRolePolicyDocument": {
                   "Version": "2012-10-17",
```

```
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {
                                "Service": [
                                    {
                                        "Fn::Sub": "states.
${AWS::Region}.amazonaws.com"
                                    }
                                ]
                            },
                            "Action": "sts:AssumeRole"
                        }
                    ]
                },
                "Path": "/",
                "Policies": [
                    {
                        "PolicyName": "StatesExecutionPolicy",
                        "PolicyDocument": {
                            "Version": "2012-10-17",
                            "Statement": [
                                {
                                    "Effect": "Allow",
                                    "Action": [
                                        "lambda:InvokeFunction"
                                    ],
                                    "Resource": "*"
                                }
                            ]
                        }
                    }
                ]
            }
        },
        "MyStateMachine": {
            "Type": "AWS::StepFunctions::StateMachine",
            "Properties": {
                "DefinitionString": {
                    "Fn::Sub": [
                        "{\n  \"Comment\": \"A Hello World example using
 an AWS Lambda function\",\n  \"StartAt\": \"HelloWorld\",\n  \"States\":
 {\n    \"HelloWorld\": {\n      \"Type\": \"Task\",\n      \"Resource\":
 \"${lambdaArn}\",\n      \"End\": true\n    }\n  }\n}",
```

```
                        {
                            "lambdaArn": {
                                "Fn::GetAtt": [
                                    "MyLambdaFunction",
                                    "Arn"
                                ]
                            }
                        }
                    ]
                },
                "RoleArn": {
                    "Fn::GetAtt": [
                        "StatesExecutionRole",
                        "Arn"
                    ]
                }
            }
        }
    }
}
```

2. Buka konsol AWS CloudFormation tersebut dan pilih Buat Tumpukan.
3. Di halaman Pilih Templat, pilih Unggah templat ke Amazon S3. Pilih file `MyStateMachine` Anda, lalu pilih Berikutnya.
4. Di halaman Tentukan Detail, masukkan Nama tumpukan, masukkan `MyStateMachine`, lalu pilih Selanjutnya.
5. Pada halaman Opsi, pilih Selanjutnya.
6. Di halaman Tinjau, pilih Saya mengakui bahwa AWS CloudFormation dapat membuat sumber daya IAM. lalu pilih Buat.

   AWS CloudFormation mulai membuat `MyStateMachine` tumpukan dan menampilkan status CREATE_IN_PROGRESS. Saat proses selesai, AWS CloudFormation menampilkan status CREATE_COMPLETE.
7. (Opsional) Untuk menampilkan sumber daya di tumpukan Anda, pilih tumpukan dan pilih tab Sumber Daya.



▼ Resources

To view detailed drift information for specific resources, visit the Drift Details page.

Logical ID	Physical ID	Type	Drift Status	Status	Status Reason
LambdaExecutionRole	MyStateMachine-LambdaExecutionRole-1[redacted]	AWS::IAM::Role	NOT_CHECKED	CREATE_COMPLETE	
MyLambdaFunction	MyStateMachine-MyLambdaFunction-[redacted]	AWS::Lambda::Function	NOT_CHECKED	CREATE_COMPLETE	
MyStateMachine	arn:aws:states:us-east-1:[redacted]:stateMachine:MyStateMachine-[redacted]	AWS::StepFunctions::State…	NOT_CHECKED	CREATE_COMPLETE	
StatesExecutionRole	MyStateMachine-StatesExecutionRole-[redacted]	AWS::IAM::Role	NOT_CHECKED	CREATE_COMPLETE	

Langkah 3: Mulai eksekusi Mesin Status

Setelah Anda membuat mesin status Lambda Anda, Anda dapat memulai eksekusi.

Untuk memulai eksekusi mesin status

1. Buka konsol Step Functions dan pilih nama mesin status yang Anda buat menggunakan AWS CloudFormation.
2. Pada halaman ***MyStateMachine-ABCDEFGHIJ1K***, pilih Eksekusi baru.

   Halaman Eksekusi baru ditampilkan.
3. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

   > **Note**
   >
   > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.
4. Pilih Mulai Eksekusi.

   Eksekusi baru mesin status Anda dimulai, dan halaman baru yang menunjukkan eksekusi Anda yang sedang berjalan akan ditampilkan.
5. (Opsional) Dalam Detail Eksekusi, tinjau stempel waktu Status eksekusi dan Dimulai dan Ditutup.
6. Untuk melihat hasil eksekusi Anda, pilih Output.

Membuat mesin Lambda negara untuk Step Functions digunakan AWS CDK

Tutorial ini menunjukkan cara membuat mesin AWS Step Functions state yang berisi AWS Lambda fungsi menggunakanAWS Cloud Development Kit (AWS CDK). AWS CDKIni adalah kerangka Infrastructure as Code (IAC) yang memungkinkan Anda mendefinisikan AWS infrastruktur menggunakan bahasa pemrograman yang lengkap. Anda dapat menulis aplikasi dalam salah satu bahasa yang CDK didukung yang berisi satu atau beberapa tumpukan. Kemudian, Anda dapat mensintesisnya ke AWS CloudFormation template dan menyebarkannya ke akun Anda AWS . Kita akan menggunakan metode ini untuk mendefinisikan mesin Step Functions state yang berisi Lambda fungsi, kemudian menggunakan AWS Management Console untuk menjalankan state machine.

Sebelum Anda memulai tutorial ini, Anda harus mengatur lingkungan AWS CDK pengembangan Anda seperti yang dijelaskan dalam Memulai Dengan AWS CDK - Prasyarat dalam Panduan Pengembang.AWS Cloud Development Kit (AWS CDK) Kemudian, instal AWS CDK dengan perintah berikut diAWS CLI:

```
npm install -g aws-cdk
```

Tutorial ini menghasilkan hasil yang sama sepertithe section called “Membuat Mesin Negara Lambda Menggunakan AWS CloudFormation”. Namun, dalam tutorial ini, AWS CDK tidak mengharuskan Anda untuk membuat IAM peran apa pun; itu AWS CDK melakukannya untuk Anda. AWS CDKVersi ini juga menyertakan Berhasil langkah untuk mengilustrasikan cara menambahkan langkah tambahan ke mesin status Anda.

> **Tip**
>
> Untuk menerapkan contoh aplikasi tanpa server yang memulai Step Functions alur kerja menggunakan AWS CDK with TypeScript ke Anda Akun AWS, lihat Modul 10 - Menerapkan dengan Workshop. AWS CDK AWS Step Functions

Topik

- Langkah 1: Siapkan proyek AWS CDK Anda
- Langkah 2: Gunakan AWS CDK untuk membuat mesin negara
- Langkah 3: Mulai eksekusi mesin negara

- Langkah 4: Bersihkan
- Langkah selanjutnya

Langkah 1: Siapkan proyek AWS CDK Anda

1. Di direktori home Anda, atau direktori lain jika Anda mau, jalankan perintah berikut untuk membuat direktori untuk AWS CDK aplikasi baru Anda.

   > **Important**
   >
   > Pastikan untuk memberi nama `step` direktori. Templat aplikasi AWS CDK menggunakan nama direktori untuk membuat nama untuk file sumber dan kelas. Jika Anda menggunakan nama yang berbeda, aplikasi Anda tidak akan cocok dengan tutorial ini.

   TypeScript

   ```
   mkdir step && cd step
   ```

   JavaScript

   ```
   mkdir step && cd step
   ```

   Python

   ```
   mkdir step && cd step
   ```

   Java

   ```
   mkdir step && cd step
   ```

   C#

   Pastikan Anda telah menginstal .NET versi 6.0 atau lebih tinggi. Untuk selengkapnya, lihat Versi yang didukung.

   ```
   mkdir step && cd step
   ```

2. Inisialisasi aplikasi dengan menggunakan perintah cdk init. Tentukan template yang diinginkan (“aplikasi”) dan bahasa pemrograman seperti yang ditunjukkan pada contoh berikut.

   TypeScript

   ```
   cdk init --language typescript
   ```

   JavaScript

   ```
   cdk init --language javascript
   ```

   Python

   ```
   cdk init --language python
   ```

   Setelah proyek diinisialisasi, aktifkan lingkungan virtual proyek dan instal AWS CDK dependensi dasar.

   ```
   source .venv/bin/activate
   python -m pip install -r requirements.txt
   ```

   Java

   ```
   cdk init --language java
   ```

   C#

   ```
   cdk init --language csharp
   ```

Langkah 2: Gunakan AWS CDK untuk membuat mesin negara

Pertama, kita akan menyajikan potongan-potongan kode individual yang mendefinisikan Lambda fungsi dan mesin Step Functions negara. Kemudian, kami akan menjelaskan cara menyatukannya di AWS CDK aplikasi Anda. Akhirnya, Anda akan melihat cara mensintesis dan menyebarkan sumber daya ini.

Untuk membuat Lambda fungsi

AWS CDKKode berikut mendefinisikan Lambda fungsi, menyediakan kode sumber inline.

TypeScript

```
const helloFunction = new lambda.Function(this, 'MyLambdaFunction', {
    code: lambda.Code.fromInline(`
          exports.handler = (event, context, callback) => {
              callback(null, "Hello World!");
          };
      `),
    runtime: lambda.Runtime.NODEJS_18_X,
    handler: "index.handler",
    timeout: cdk.Duration.seconds(3)
});
```

JavaScript

```
const helloFunction = new lambda.Function(this, 'MyLambdaFunction', {
    code: lambda.Code.fromInline(`
          exports.handler = (event, context, callback) => {
              callback(null, "Hello World!");
          };
      `),
    runtime: lambda.Runtime.NODEJS_18_X,
    handler: "index.handler",
    timeout: cdk.Duration.seconds(3)
});
```

Python

```
hello_function = lambda_.Function(
            self, "MyLambdaFunction",
            code=lambda_.Code.from_inline("""
            exports.handler = (event, context, callback) => {
                callback(null, "Hello World!");
                }"""),
                runtime=lambda_.Runtime.NODEJS_18_X,
                handler="index.handler",
                timeout=Duration.seconds(25))
```

Java

```
final Function helloFunction = Function.Builder.create(this, "MyLambdaFunction")
        .code(Code.fromInline(
                "exports.handler = (event, context, callback) => { callback(null,
 'Hello World!' );}"))
        .runtime(Runtime.NODEJS_18_X)
        .handler("index.handler")
        .timeout(Duration.seconds(25))
        .build();
```

C#

```
var helloFunction = new Function(this, "MyLambdaFunction", new FunctionProps
{
    Code = Code.FromInline(@"`
      exports.handler = (event, context, callback) => {
        callback(null, 'Hello World!');
      }"),
    Runtime = Runtime.NODEJS_12_X,
    Handler = "index.handler",
    Timeout = Duration.Seconds(25)
});
```

Anda dapat melihat dalam kode contoh singkat ini:

- Nama logis fungsi, `MyLambdaFunction`.
- Kode sumber untuk fungsi tersebut, disematkan sebagai string dalam kode sumber aplikasi AWS CDK.
- Atribut fungsi lainnya, seperti runtime yang akan digunakan (Node 18.x), titik masuk fungsi, dan batas waktu.

Untuk membuat mesin status

Mesin status kami memiliki dua status: tugas Lambda fungsi, dan Berhasil status. Fungsi ini mengharuskan kita membuat sebuah Step Functions the section called “Tugas” yang memanggil fungsi kita. Status Tugas ini digunakan sebagai langkah pertama dalam mesin negara. Status sukses ditambahkan ke mesin status menggunakan `next()` metode status Tugas. Kode berikut

pertama memanggil fungsi bernama`MyLambdaTask`, kemudian menggunakan `next()` metode untuk mendefinisikan status sukses bernama`GreetedWorld`.

TypeScript

```
const stateMachine = new sfn.StateMachine(this, 'MyStateMachine', {
  definition: new tasks.LambdaInvoke(this, "MyLambdaTask", {
    lambdaFunction: helloFunction
  }).next(new sfn.Succeed(this, "GreetedWorld"))
});
```

JavaScript

```
const stateMachine = new sfn.StateMachine(this, 'MyStateMachine', {
  definition: new tasks.LambdaInvoke(this, "MyLambdaTask", {
    lambdaFunction: helloFunction
  }).next(new sfn.Succeed(this, "GreetedWorld"))
});
```

Python

```
state_machine = sfn.StateMachine(
                                 self, "MyStateMachine",
                                 definition=tasks.LambdaInvoke(
                                 self, "MyLambdaTask",
                                 lambda_function=hello_function)
                                 .next(sfn.Succeed(self, "GreetedWorld")))
```

Java

```
final StateMachine stateMachine = StateMachine.Builder.create(this,
 "MyStateMachine")
        .definition(LambdaInvoke.Builder.create(this, "MyLambdaTask")
            .lambdaFunction(helloFunction)
            .build()
            .next(new Succeed(this, "GreetedWorld")))
        .build();
```

C#

```
var stateMachine = new StateMachine(this, "MyStateMachine", new StateMachineProps {
```

```
    DefinitionBody = DefinitionBody.FromChainable(new LambdaInvoke(this,
 "MyLambdaTask", new LambdaInvokeProps
    {
        LambdaFunction = helloFunction
    })
    .Next(new Succeed(this, "GreetedWorld")))
});
```

Untuk membangun dan men-deploy aplikasi AWS CDK

Dalam AWS CDK proyek Anda yang baru dibuat, edit file yang berisi definisi tumpukan agar terlihat seperti kode contoh berikut. Anda akan mengenali definisi Lambda fungsi dan mesin Step Functions status dari bagian sebelumnya.

1. Perbarui tumpukan seperti yang ditunjukkan pada contoh berikut.

   TypeScript

   Perbarui `lib/step-stack.ts` dengan kode berikut.

```
import * as cdk from 'aws-cdk-lib';
import * as lambda from 'aws-cdk-lib/aws-lambda';
import * as sfn from 'aws-cdk-lib/aws-stepfunctions';
import * as tasks from 'aws-cdk-lib/aws-stepfunctions-tasks';

export class StepStack extends cdk.Stack {
  constructor(app: cdk.App, id: string) {
    super(app, id);

    const helloFunction = new lambda.Function(this, 'MyLambdaFunction', {
      code: lambda.Code.fromInline(`
          exports.handler = (event, context, callback) => {
              callback(null, "Hello World!");
          };
      `),
      runtime: lambda.Runtime.NODEJS_18_X,
      handler: "index.handler",
      timeout: cdk.Duration.seconds(3)
    });

    const stateMachine = new sfn.StateMachine(this, 'MyStateMachine', {
      definition: new tasks.LambdaInvoke(this, "MyLambdaTask", {
```

```
        lambdaFunction: helloFunction
      }).next(new sfn.Succeed(this, "GreetedWorld"))
    });
  }
}
```

JavaScript

Perbarui `lib/step-stack.js` dengan kode berikut.

```
import * as cdk from 'aws-cdk-lib';
import * as lambda from 'aws-cdk-lib/aws-lambda';
import * as sfn from 'aws-cdk-lib/aws-stepfunctions';
import * as tasks from 'aws-cdk-lib/aws-stepfunctions-tasks';

export class StepStack extends cdk.Stack {
  constructor(app, id) {
    super(app, id);

    const helloFunction = new lambda.Function(this, 'MyLambdaFunction', {
      code: lambda.Code.fromInline(`
          exports.handler = (event, context, callback) => {
              callback(null, "Hello World!");
          };
      `),
      runtime: lambda.Runtime.NODEJS_18_X,
      handler: "index.handler",
      timeout: cdk.Duration.seconds(3)
    });

    const stateMachine = new sfn.StateMachine(this, 'MyStateMachine', {
      definition: new tasks.LambdaInvoke(this, "MyLambdaTask", {
        lambdaFunction: helloFunction
      }).next(new sfn.Succeed(this, "GreetedWorld"))
    });
  }
}
```

Python

Perbarui `step/step_stack.py` dengan kode berikut.

```
from aws_cdk import (
```

```
    Duration,
    Stack,
    aws_stepfunctions as sfn,
    aws_stepfunctions_tasks as tasks,
    aws_lambda as lambda_
)
class StepStack(Stack):

    def __init__(self, scope: Construct, construct_id: str, **kwargs) -> None:
        super().__init__(scope, construct_id, **kwargs)

        hello_function = lambda_.Function(
            self, "MyLambdaFunction",
            code=lambda_.Code.from_inline("""
            exports.handler = (event, context, callback) => {
                callback(null, "Hello World!");
                }"""),
                runtime=lambda_.Runtime.NODEJS_18_X,
                handler="index.handler",
                timeout=Duration.seconds(25))

        state_machine = sfn.StateMachine(
            self, "MyStateMachine",
            definition=tasks.LambdaInvoke(
            self, "MyLambdaTask",
            lambda_function=hello_function)
            .next(sfn.Succeed(self, "GreetedWorld")))
```

Java

Perbarui `src/main/java/com.myorg/StepStack.java` dengan kode berikut.

```
package com.myorg;

import software.constructs.Construct;
import software.amazon.awscdk.Stack;
import software.amazon.awscdk.StackProps;
import software.amazon.awscdk.Duration;
import software.amazon.awscdk.services.lambda.Code;
import software.amazon.awscdk.services.lambda.Function;
import software.amazon.awscdk.services.lambda.Runtime;
import software.amazon.awscdk.services.stepfunctions.StateMachine;
import software.amazon.awscdk.services.stepfunctions.Succeed;
```

```
import software.amazon.awscdk.services.stepfunctions.tasks.LambdaInvoke;

public class StepStack extends Stack {
    public StepStack(final Construct scope, final String id) {
        this(scope, id, null);
    }

    public StepStack(final Construct scope, final String id, final StackProps
 props) {
        super(scope, id, props);

        final Function helloFunction = Function.Builder.create(this,
 "MyLambdaFunction")
                .code(Code.fromInline(
                        "exports.handler = (event, context, callback) =>
 { callback(null, 'Hello World!' );}"))
                .runtime(Runtime.NODEJS_18_X)
                .handler("index.handler")
                .timeout(Duration.seconds(25))
                .build();

        final StateMachine stateMachine = StateMachine.Builder.create(this,
 "MyStateMachine")
                .definition(LambdaInvoke.Builder.create(this, "MyLambdaTask")
                        .lambdaFunction(helloFunction)
                        .build()
                        .next(new Succeed(this, "GreetedWorld")))
                .build();
    }
}
```

C#

Perbarui `scr/Step/StepStack.cs` dengan kode berikut.

```
using Amazon.CDK;
using Constructs;
using Amazon.CDK.AWS.Lambda;
using Amazon.CDK.AWS.StepFunctions;
using Amazon.CDK.AWS.StepFunctions.Tasks;

namespace Step
{
```

```
    public class StepStack : Stack
    {
        internal StepStack(Construct scope, string id, IStackProps props =
null) : base(scope, id, props)
        {
            var helloFunction = new Function(this, "MyLambdaFunction", new
FunctionProps
            {
                Code = Code.FromInline(@"exports.handler = (event, context,
callback) => {
                    callback(null, 'Hello World!');
                }"),
                Runtime = Runtime.NODEJS_18_X,
                Handler = "index.handler",
                Timeout = Duration.Seconds(25)
            });

            var stateMachine = new StateMachine(this, "MyStateMachine", new
StateMachineProps
            {
                DefinitionBody = DefinitionBody.FromChainable(new
LambdaInvoke(this, "MyLambdaTask", new LambdaInvokeProps
                {
                    LambdaFunction = helloFunction
                })
                .Next(new Succeed(this, "GreetedWorld")))
            });
        }
    }
}
```

2. Simpan file sumber, lalu jalankan `cdk synth` perintah di direktori utama aplikasi.

   AWS CDKmenjalankan aplikasi dan mensintesis AWS CloudFormation template darinya. AWS CDKkemudian menampilkan template.

   > **Note**
   >
   > Jika Anda biasa TypeScript membuat AWS CDK proyek Anda, menjalankan `cdk synth` perintah dapat mengembalikan kesalahan berikut.
   >
   > ```
   > TSError: # Unable to compile TypeScript:
   > ```

```
bin/step.ts:7:33 - error TS2554: Expected 2 arguments, but got 3.
```

Ubah `bin/step.ts` file seperti yang ditunjukkan pada contoh berikut untuk mengatasi kesalahan ini.

```
#!/usr/bin/env node
import 'source-map-support/register';
import * as cdk from 'aws-cdk-lib';
import { StepStack } from '../lib/step-stack';

const app = new cdk.App();
new StepStack(app, 'StepStack');
app.synth();
```

3. Untuk menerapkan fungsi Lambda dan mesin status Step Functions ke akun AWS Anda, masalah. `cdk deploy` Anda akan diminta untuk menyetujui kebijakan IAM yang AWS CDK telah dihasilkan.

Langkah 3: Mulai eksekusi mesin negara

Setelah Anda membuat mesin negara Anda, Anda dapat memulai eksekusi.

Untuk memulai eksekusi mesin status

1. Buka Konsol Step Functions lalu pilih nama mesin status yang Anda buat dengan menggunakan AWS CDK.
2. Pada halaman mesin status, pilih Mulai eksekusi.

   Kotak dialog Mulai eksekusi ditampilkan.
3. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

> **Note**
>
> Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

4. Pilih Mulai Eksekusi.

   Eksekusi state machine Anda dimulai, dan halaman baru yang menunjukkan eksekusi yang sedang berjalan ditampilkan.

5. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

   Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Langkah 4: Bersihkan

Setelah Anda menguji mesin status Anda, kami sarankan Anda menghapus mesin status dan fungsi Lambda terkait untuk membebaskan sumber daya di mesin Anda. Akun AWS Jalankan `cdk destroy` perintah di direktori utama aplikasi Anda untuk menghapus mesin status Anda.

Langkah selanjutnya

Untuk mempelajari lebih lanjut tentang mengembangkan AWS infrastruktur menggunakanAWS CDK, lihat Panduan AWS CDK Pengembang.

Untuk informasi tentang menulis aplikasi AWS CDK dalam bahasa pilihan Anda, lihat:

TypeScript

Bekerja dengan AWS CDK di TypeScript

JavaScript

Bekerja dengan AWS CDK di JavaScript

Python

Bekerja dengan AWS CDK Python

Java

Bekerja dengan AWS CDK di Jawa

C#

Bekerja dengan AWS CDK di C #

Untuk informasi selengkapnya tentang modul AWS Construct Library yang digunakan dalam tutorial ini, lihat ikhtisar Referensi AWS CDK API berikut:

- aws-lambda
- aws-stepfunctions
- aws-stepfunctions-tasks

Membuat API REST API Gateway dengan Mesin Status Ekspres Sinkron Menggunakan AWS CDK

Tutorial ini menunjukkan cara membuat API Gateway REST API dengan Synchronous Express State Machine sebagai integrasi backend menggunakan. AWS Cloud Development Kit (AWS CDK) Tutorial ini akan menggunakan `StepFunctionsRestApi` konstruksi untuk menghubungkan State Machine ke API Gateway. `StepFunctionsRestApi`Konstruksi akan menyiapkan pemetaan input/output default dan API Gateway REST API, dengan izin yang diperlukan dan metode HTTP "APAPUN". AWS CDK Ini adalah kerangka Infrastructure as Code (IAC) yang memungkinkan Anda mendefinisikan AWS infrastruktur menggunakan bahasa pemrograman yang lengkap. Anda menulis aplikasi dalam salah satu bahasa yang didukung CDK, berisi satu atau beberapa tumpukan, lalu mensintesiskannya ke AWS CloudFormation templat dan menerapkannya ke akun Anda. AWS Kita akan menggunakannya untuk mendefinisikan API Gateway REST API, yang terintegrasi dengan Synchronous Express State Machine sebagai backend, lalu gunakan AWS Management Console untuk memulai eksekusi.

Sebelum memulai tutorial ini, siapkan lingkungan AWS CDK pengembangan Anda seperti yang dijelaskan dalam Memulai Dengan AWS CDK - Prasyarat, lalu instal dengan menerbitkan: AWS CDK

```
npm install -g aws-cdk
```

Topik

- Langkah 1: Siapkan Proyek AWS CDK Anda
- Langkah 2: Gunakan AWS CDK untuk membuat API Gateway REST API dengan integrasi backend Synchronous Express State Machine

- Langkah 3: Uji API Gateway
- Langkah 4: Bersihkan

Langkah 1: Siapkan Proyek AWS CDK Anda

Pertama, buat direktori untuk AWS CDK aplikasi baru Anda dan inisialisasi proyek.

TypeScript

```
mkdir stepfunctions-rest-api
cd stepfunctions-rest-api
cdk init --language typescript
```

JavaScript

```
mkdir stepfunctions-rest-api
cd stepfunctions-rest-api
cdk init --language javascript
```

Python

```
mkdir stepfunctions-rest-api
cd stepfunctions-rest-api
cdk init --language python
```

Setelah proyek diinisialisasi, aktifkan lingkungan virtual proyek dan instal AWS CDK dependensi dasar.

```
source .venv/bin/activate
python -m pip install -r requirements.txt
```

Java

```
mkdir stepfunctions-rest-api
cd stepfunctions-rest-api
cdk init --language java
```

C#

```
mkdir stepfunctions-rest-api
```

```
cd stepfunctions-rest-api
cdk init --language csharp
```

Go

```
mkdir stepfunctions-rest-api
cd stepfunctions-rest-api
cdk init --language go
```

> **Note**
>
> Pastikan untuk memberi nama `stepfunctions-rest-api` direktori. Template AWS CDK aplikasi menggunakan nama direktori untuk menghasilkan nama untuk file sumber dan kelas. Jika Anda menggunakan nama yang berbeda, aplikasi Anda tidak akan cocok dengan tutorial ini.

Sekarang instal modul pustaka konstruksi untuk AWS Step Functions dan Amazon API Gateway.

TypeScript

```
npm install @aws-cdk/aws-stepfunctions @aws-cdk/aws-apigateway
```

JavaScript

```
npm install @aws-cdk/aws-stepfunctions @aws-cdk/aws-apigateway
```

Python

```
python -m pip install aws-cdk.aws-stepfunctions
python -m pip install aws-cdk.aws-apigateway
```

Java

Edit `pom.xml` proyek untuk menambahkan dependensi berikut di dalam kontainer `<dependencies>` yang sudah ada.

```
        <dependency>
            <groupId>software.amazon.awscdk</groupId>
```

```
            <artifactId>stepfunctions</artifactId>
            <version>${cdk.version}</version>
        </dependency>
        <dependency>
            <groupId>software.amazon.awscdk</groupId>
            <artifactId>apigateway</artifactId>
            <version>${cdk.version}</version>
        </dependency>
```

Maven menginstal dependensi ini secara otomatis saat Anda membangun aplikasi berikutnya. Untuk membangun, terbitkan `mvn compile` atau gunakan perintah Bangun IDE Java Anda.

C#

```
dotnet add src/StepfunctionsRestApi package Amazon.CDK.AWS.Stepfunctions
dotnet add src/StepfunctionsRestApi package Amazon.CDK.AWS.APIGateway
```

Anda juga dapat menginstal paket yang ditunjukkan menggunakan NuGet GUI Visual Studio, tersedia melalui Tools > NuGet Package Manager > Manage NuGet Packages for Solution.

Setelah menginstal modul, Anda dapat menggunakannya di AWS CDK aplikasi Anda dengan mengimpor paket-paket berikut.

TypeScript

```
@aws-cdk/aws-stepfunctions
@aws-cdk/aws-apigateway
```

JavaScript

```
@aws-cdk/aws-stepfunctions
@aws-cdk/aws-apigateway
```

Python

```
aws_cdk.aws_stepfunctions
aws_cdk.aws_apigateway
```

Java

```
software.amazon.awscdk.services.apigateway.StepFunctionsRestApi
```

```
software.amazon.awscdk.services.stepfunctions.Pass
software.amazon.awscdk.services.stepfunctions.StateMachine
software.amazon.awscdk.services.stepfunctions.StateMachineType
```

C#

```
Amazon.CDK.AWS.StepFunctions
Amazon.CDK.AWS.APIGateway
```

Go

Tambahkan yang berikut ini ke `import` dalam`stepfunctions-rest-api.go`.

```
"github.com/aws/aws-cdk-go/awscdk/awsapigateway"
"github.com/aws/aws-cdk-go/awscdk/awsstepfunctions"
```

Langkah 2: Gunakan AWS CDK untuk membuat API Gateway REST API dengan integrasi backend Synchronous Express State Machine

Pertama, kami akan menyajikan potongan kode individual yang mendefinisikan Synchronous Express State Machine dan API Gateway REST API, lalu menjelaskan cara menyatukannya ke dalam AWS CDK aplikasi Anda. Kemudian Anda akan melihat cara mensintesis dan men-deploy sumber daya ini.

> **Note**
>
> Mesin Negara yang akan kami tunjukkan di sini akan menjadi Mesin Negara sederhana dengan `Pass` status.

Untuk membuat Mesin Negara Ekspres

Ini adalah AWS CDK kode yang mendefinisikan mesin negara sederhana dengan `Pass` status.

TypeScript

```
const machineDefinition = new stepfunctions.Pass(this, 'PassState', {
    result: {value:"Hello!"},
})
```

```
const stateMachine = new stepfunctions.StateMachine(this, 'MyStateMachine', {
    definition: machineDefinition,
    stateMachineType: stepfunctions.StateMachineType.EXPRESS,
});
```

JavaScript

```
const machineDefinition = new sfn.Pass(this, 'PassState', {
    result: {value:"Hello!"},
})

const stateMachine = new sfn.StateMachine(this, 'MyStateMachine', {
    definition: machineDefinition,
    stateMachineType: stepfunctions.StateMachineType.EXPRESS,
});
```

Python

```
machine_definition = sfn.Pass(self,"PassState",
                        result = sfn.Result("Hello"))

state_machine = sfn.StateMachine(self, 'MyStateMachine',
        definition = machine_definition,
        state_machine_type = sfn.StateMachineType.EXPRESS)
```

Java

```
Pass machineDefinition = Pass.Builder.create(this, "PassState")
                        .result(Result.fromString("Hello"))
                        .build();

StateMachine stateMachine = StateMachine.Builder.create(this, "MyStateMachine")
                            .definition(machineDefinition)
                            .stateMachineType(StateMachineType.EXPRESS)
                            .build();
```

C#

```
var machineDefinition = new Pass(this, "PassState", new PassProps
{
    Result = Result.FromString("Hello")
});
```

```
var stateMachine = new StateMachine(this, "MyStateMachine", new StateMachineProps
{
    Definition = machineDefinition,
    StateMachineType = StateMachineType.EXPRESS
});
```

Go

```
var machineDefinition = awsstepfunctions.NewPass(stack, jsii.String("PassState"),
 &awsstepfunctions.PassProps
{
    Result: awsstepfunctions.NewResult(jsii.String("Hello")),
})

var stateMachine = awsstepfunctions.NewStateMachine(stack,
 jsii.String("StateMachine"), &awsstepfunctions.StateMachineProps
{
    Definition: machineDefinition,
    StateMachineType: awsstepfunctions.StateMachineType_EXPRESS,
})
```

Anda dapat melihat dalam cuplikan singkat ini:

- Definisi mesin bernama`PassState`, yang merupakan `Pass` Negara.
- Nama logis dari State Machine, `MyStateMachine`
- Definisi mesin digunakan sebagai definisi State Machine.
- Jenis Mesin Negara diatur sebagai `EXPRESS` karena hanya `StepFunctionsRestApi` akan mengizinkan mesin status Synchronous Express.

Untuk membuat API API Gateway REST API menggunakan **`StepFunctionsRestApi`** build

Kami akan menggunakan `StepFunctionsRestApi` konstruksi untuk membuat API Gateway REST API dengan izin yang diperlukan dan pemetaan input/output default.

TypeScript

```
const api = new apigateway.StepFunctionsRestApi(this,
```

```
  'StepFunctionsRestApi', { stateMachine: stateMachine });
```

JavaScript

```
const api = new apigateway.StepFunctionsRestApi(this,
  'StepFunctionsRestApi', { stateMachine: stateMachine });
```

Python

```
api = apigw.StepFunctionsRestApi(self, "StepFunctionsRestApi",
                            state_machine = state_machine)
```

Java

```
StepFunctionsRestApi api = StepFunctionsRestApi.Builder.create(this,
 "StepFunctionsRestApi")
                           .stateMachine(stateMachine)
                           .build();
```

C#

```
var api = new StepFunctionsRestApi(this, "StepFunctionsRestApi", new
 StepFunctionsRestApiProps
{
    StateMachine = stateMachine
});
```

Go

```
awsapigateway.NewStepFunctionsRestApi(stack, jsii.String("StepFunctionsRestApi"),
 &awsapigateway.StepFunctionsRestApiProps
{
    StateMachine = stateMachine,
})
```

Untuk membangun dan men-deploy aplikasi AWS CDK

Dalam AWS CDK proyek yang Anda buat, edit file yang berisi definisi tumpukan agar terlihat seperti kode di bawah ini. Anda akan mengenali definisi mesin status Step Functions dan API Gateway dari atas.

TypeScript

Perbarui `lib/stepfunctions-rest-api-stack.ts` untuk membaca sebagai berikut.

```
import * as cdk from 'aws-cdk-lib';
import * as stepfunctions from 'aws-cdk-lib/aws-stepfunctions'
import * as apigateway from 'aws-cdk-lib/aws-apigateway';


export class StepfunctionsRestApiStack extends cdk.Stack {
    constructor(scope: cdk.App, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    const machineDefinition = new stepfunctions.Pass(this, 'PassState', {
        result: {value:"Hello!"},
    });

    const stateMachine = new stepfunctions.StateMachine(this, 'MyStateMachine', {
        definition: machineDefinition,
        stateMachineType: stepfunctions.StateMachineType.EXPRESS,
    });

    const api = new apigateway.StepFunctionsRestApi(this,
        'StepFunctionsRestApi', { stateMachine: stateMachine });
```

JavaScript

Perbarui `lib/stepfunctions-rest-api-stack.js` untuk membaca sebagai berikut.

```
const cdk = require('@aws-cdk/core');
const stepfunctions = require('@aws-cdk/aws-stepfunctions');
const apigateway = require('@aws-cdk/aws-apigateway');


class StepfunctionsRestApiStack extends cdk.Stack {
    constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    const machineDefinition = new stepfunctions.Pass(this, "PassState", {
        result: {value:"Hello!"},
    })

    const stateMachine = new sfn.StateMachine(this, 'MyStateMachine', {
        definition: machineDefinition,
```

```
        stateMachineType: stepfunctions.StateMachineType.EXPRESS,
    });

    const api = new apigateway.StepFunctionsRestApi(this,
        'StepFunctionsRestApi', { stateMachine: stateMachine });

    }
}

module.exports = { StepStack }
```

Python

Perbarui `stepfunctions_rest_api/stepfunctions_rest_api_stack.py` untuk membaca sebagai berikut.

```
from aws_cdk import App, Stack
from constructs import Construct
from aws_cdk import aws_stepfunctions as sfn
from aws_cdk import aws_apigateway as apigw

class StepfunctionsRestApiStack(Stack):

    def __init__(self, scope: Construct, construct_id: str, **kwargs) -> None:
        super().__init__(scope, construct_id, **kwargs)


        machine_definition = sfn.Pass(self,"PassState",
                                result = sfn.Result("Hello"))

        state_machine = sfn.StateMachine(self, 'MyStateMachine',
                definition = machine_definition,
                state_machine_type = sfn.StateMachineType.EXPRESS)

        api = apigw.StepFunctionsRestApi(self,
                    "StepFunctionsRestApi",
                    state_machine = state_machine)
```

Java

Perbarui `src/main/java/com.myorg/StepfunctionsRestApiStack.java` untuk membaca sebagai berikut.

```
package com.myorg;


import software.amazon.awscdk.core.Construct;
import software.amazon.awscdk.core.Stack;
import software.amazon.awscdk.core.StackProps;
import software.amazon.awscdk.services.stepfunctions.Pass;
import software.amazon.awscdk.services.stepfunctions.StateMachine;
import software.amazon.awscdk.services.stepfunctions.StateMachineType;
import software.amazon.awscdk.services.apigateway.StepFunctionsRestApi;

public class StepfunctionsRestApiStack extends Stack {
    public StepfunctionsRestApiStack(final Construct scope, final String id) {
        this(scope, id, null);
    }

    public StepfunctionsRestApiStack(final Construct scope, final String id, final
 StackProps props) {
        super(scope, id, props);

        Pass machineDefinition = Pass.Builder.create(this, "PassState")
                                .result(Result.fromString("Hello"))
                                .build();

        StateMachine stateMachine = StateMachine.Builder.create(this,
 "MyStateMachine")
                                    .definition(machineDefinition)
                                    .stateMachineType(StateMachineType.EXPRESS)
                                    .build();

        StepFunctionsRestApi api = StepFunctionsRestApi.Builder.create(this,
 "StepFunctionsRestApi")
                                   .stateMachine(stateMachine)
                                   .build();

    }
}
```

C#

Perbarui `src/StepfunctionsRestApi/StepfunctionsRestApiStack.cs` untuk membaca sebagai berikut.

```
using Amazon.CDK;
using Amazon.CDK.AWS.StepFunctions;
using Amazon.CDK.AWS.APIGateway;

namespace StepfunctionsRestApi
{
    public class StepfunctionsRestApiStack : Stack
    {
        internal StepfunctionsRestApi(Construct scope, string id, IStackProps props
 = null) : base(scope, id, props)
        {
            var machineDefinition = new Pass(this, "PassState", new PassProps
            {
                Result = Result.FromString("Hello")
            });

            var stateMachine = new StateMachine(this, "MyStateMachine", new
 StateMachineProps
            {
                Definition = machineDefinition,
                StateMachineType = StateMachineType.EXPRESS
            });

            var api = new StepFunctionsRestApi(this, "StepFunctionsRestApi", new
 StepFunctionsRestApiProps
            {
                StateMachine = stateMachine
            });

        }
    }
}
```

Go

Perbarui `stepfunctions-rest-api.go` untuk membaca sebagai berikut.

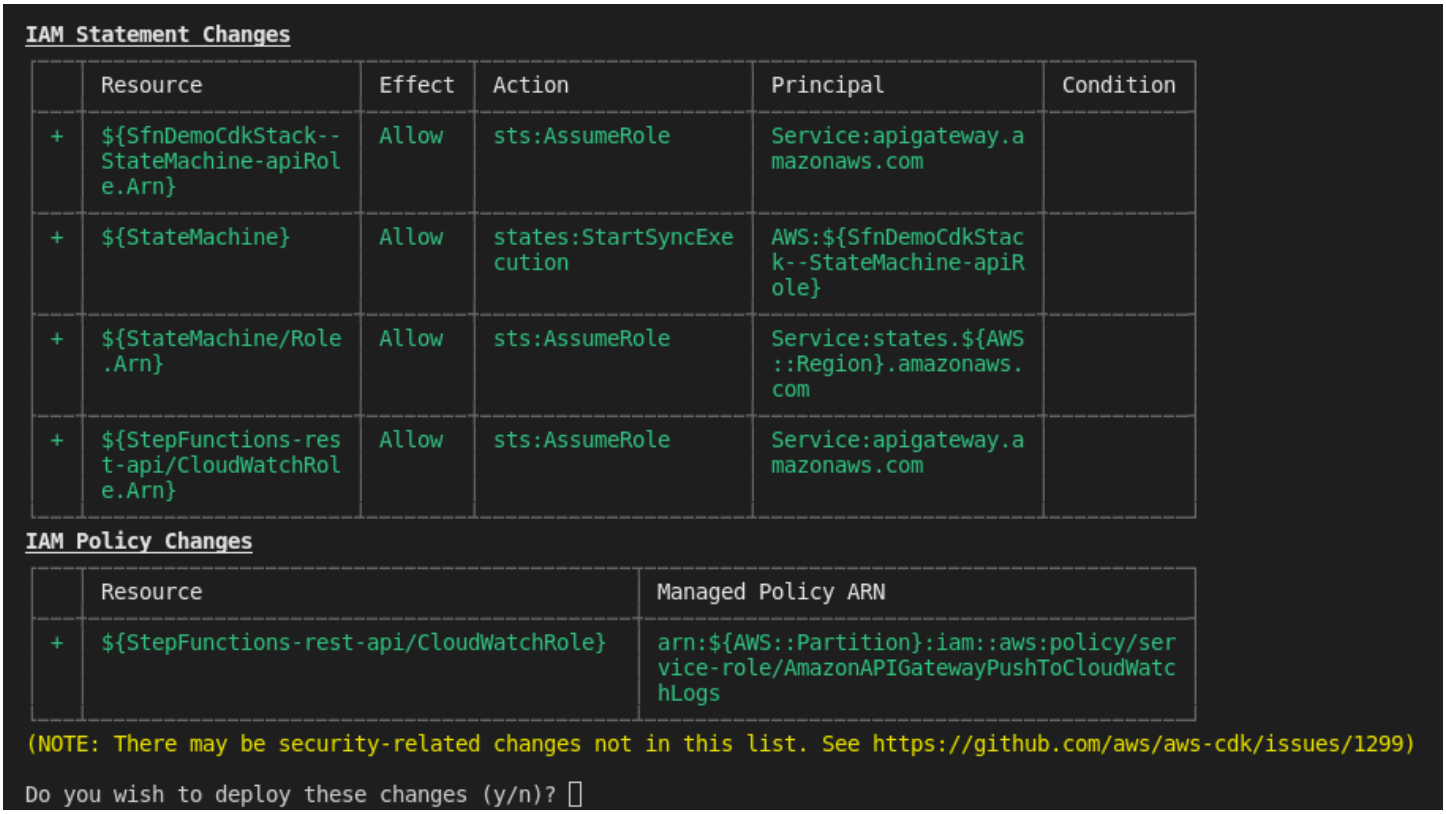
```
package main
import (
    "github.com/aws/aws-cdk-go/awscdk"
    "github.com/aws/aws-cdk-go/awscdk/awsapigateway"
    "github.com/aws/aws-cdk-go/awscdk/awsstepfunctions"
    "github.com/aws/constructs-go/constructs/v3"
```

```
    "github.com/aws/jsii-runtime-go"
)


type StepfunctionsRestApiGoStackProps struct {
    awscdk.StackProps
}

func NewStepfunctionsRestApiGoStack(scope constructs.Construct, id string, props
 *StepfunctionsRestApiGoStackProps) awscdk.Stack {
    var sprops awscdk.StackProps
    if props != nil {
        sprops = props.StackProps
    }
    stack := awscdk.NewStack(scope, &id, &sprops)

    // The code that defines your stack goes here
    var machineDefinition = awsstepfunctions.NewPass(stack,
 jsii.String("PassState"), &awsstepfunctions.PassProps
    {
        Result: awsstepfunctions.NewResult(jsii.String("Hello")),
    })

    var stateMachine = awsstepfunctions.NewStateMachine(stack,
 jsii.String("StateMachine"), &awsstepfunctions.StateMachineProps{
        Definition: machineDefinition,
        StateMachineType: awsstepfunctions.StateMachineType_EXPRESS,
    });

    awsapigateway.NewStepFunctionsRestApi(stack,
 jsii.String("StepFunctionsRestApi"), &awsapigateway.StepFunctionsRestApiProps{
        StateMachine = stateMachine,
    })

    return stack
}

func main() {
    app := awscdk.NewApp(nil)

    NewStepfunctionsRestApiGoStack(app, "StepfunctionsRestApiGoStack",
 &StepfunctionsRestApiGoStackProps{
        awscdk.StackProps{
            Env: env(),
```

```
        },
    })

    app.Synth(nil)
}

// env determines the AWS environment (account+region) in which our stack is to
// be deployed. For more information see: https://docs.aws.amazon.com/cdk/latest/
guide/environments.html
func env() *awscdk.Environment {
    // If unspecified, this stack will be "environment-agnostic".
    // Account/Region-dependent features and context lookups will not work, but a
    // single synthesized template can be deployed anywhere.
    //---------------------------------------------------------------------------
    return nil

    // Uncomment if you know exactly what account and region you want to deploy
    // the stack to. This is the recommendation for production stacks.
    //---------------------------------------------------------------------------
    // return &awscdk.Environment{
    //  Account: jsii.String("123456789012"),
    //  Region:  jsii.String("us-east-1"),
    // }

    // Uncomment to specialize this stack for the AWS Account and Region that are
    // implied by the current CLI configuration. This is recommended for dev
    // stacks.
    //---------------------------------------------------------------------------
    // return &awscdk.Environment{
    //  Account: jsii.String(os.Getenv("CDK_DEFAULT_ACCOUNT")),
    //  Region:  jsii.String(os.Getenv("CDK_DEFAULT_REGION")),
    // }
}
```

Simpan file sumber, lalu terbitkan `cdk synth` di direktori utama aplikasi. AWS CDK Menjalankan aplikasi dan mensintesis AWS CloudFormation template darinya, lalu menampilkan template.

Untuk benar-benar menerapkan Amazon API Gateway dan mesin AWS Step Functions status ke akun AWS Anda, masalah`cdk deploy`. Anda akan diminta untuk menyetujui kebijakan IAM yang AWS CDK telah dihasilkan. Kebijakan yang dibuat akan terlihat seperti ini:



IAM Statement Changes

	Resource	Effect	Action	Principal	Condition
+	${SfnDemoCdkStack--StateMachine-apiRole.Arn}	Allow	sts:AssumeRole	Service:apigateway.amazonaws.com	
+	${StateMachine}	Allow	states:StartSyncExecution	AWS:${SfnDemoCdkStack--StateMachine-apiRole}	
+	${StateMachine/Role.Arn}	Allow	sts:AssumeRole	Service:states.${AWS::Region}.amazonaws.com	
+	${StepFunctions-rest-api/CloudWatchRole.Arn}	Allow	sts:AssumeRole	Service:apigateway.amazonaws.com	

IAM Policy Changes

	Resource	Managed Policy ARN
+	${StepFunctions-rest-api/CloudWatchRole}	arn:${AWS::Partition}:iam::aws:policy/service-role/AmazonAPIGatewayPushToCloudWatchLogs

(NOTE: There may be security-related changes not in this list. See https://github.com/aws/aws-cdk/issues/1299)

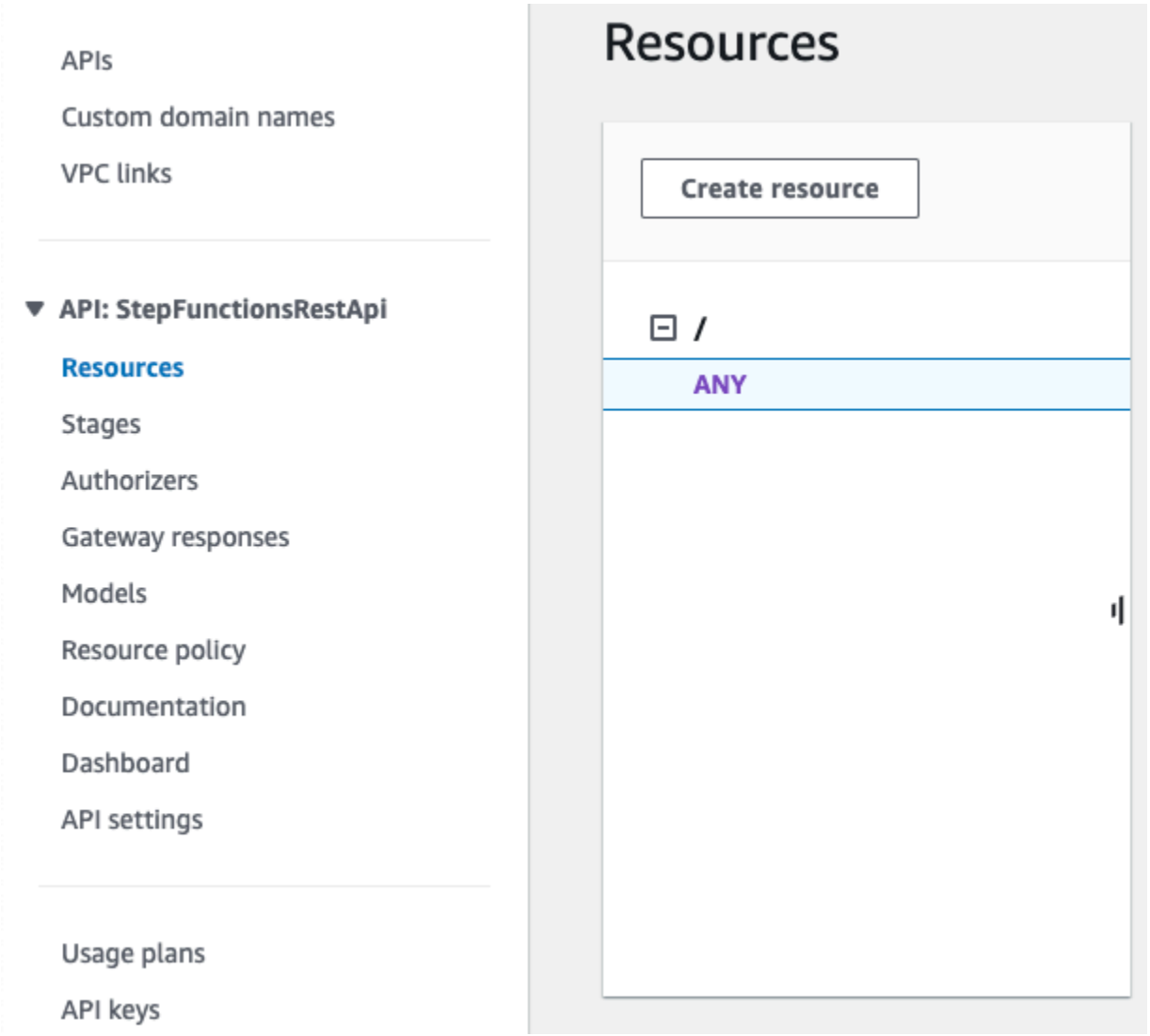
Do you wish to deploy these changes (y/n)?

Langkah 3: Uji API Gateway

Setelah Anda membuat API Gateway REST API dengan Synchronous Express State Machine sebagai integrasi backend, Anda dapat menguji API Gateway.

Untuk menguji API Gateway yang diterapkan menggunakan konsol API Gateway

1. Buka konsol Amazon API Gateway dan masuk.
2. Pilih REST API Anda bernama`StepFunctionsRestApi`.
3. Di panel Resources, pilih `ANY` metode.



4. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
5. Untuk Metode, pilih POST.
6. Untuk badan Permintaan, salin parameter permintaan berikut.

```
{
    "key": "Hello"
}
```

7. Pilih Uji. Informasi berikut akan ditampilkan:

   - Permintaan adalah jalur sumber daya yang dipanggil untuk metode.
   - Status adalah kode status HTTP respon.

- Latensi adalah waktu antara penerimaan permintaan dari penelepon dan respons yang dikembalikan.
- Response body adalah badan respon HTTP.
- Response header adalah header respon HTTP.
- Log menunjukkan simulasi entri Amazon CloudWatch Logs yang akan ditulis jika metode ini dipanggil di luar konsol API Gateway.

> **Note**
>
> Meskipun entri CloudWatch Log disimulasikan, hasil pemanggilan metode adalah nyata.

Output tubuh Response harus seperti ini:

```
"Hello"
```

> **Tip**
>
> Coba API Gateway dengan metode yang berbeda dan input yang tidak valid untuk melihat keluaran kesalahan. Anda mungkin ingin mengubah mesin status untuk mencari kunci tertentu dan selama pengujian memberikan kunci yang salah untuk gagal eksekusi State Machine dan menghasilkan pesan kesalahan dalam output badan Response.

Untuk menguji API yang diterapkan menggunakan cURL

1. Buka jendela terminal.
2. Salin perintah cURL berikut dan tempelkan ke jendela terminal, ganti `<api-id>` dengan API ID API Anda dan `<region>` dengan wilayah tempat API Anda digunakan.

```
curl -X POST\
 'https://<api-id>.execute-api.<region>.amazonaws.com/prod' \
 -d '{"key":"Hello"}' \
 -H 'Content-Type: application/json'
```

Output Response Body harus seperti ini:

```
"Hello"
```

> **Tip**
>
> Coba API Gateway dengan metode yang berbeda dan input yang tidak valid untuk melihat keluaran kesalahan. Anda mungkin ingin mengubah mesin status untuk mencari kunci tertentu dan selama pengujian memberikan kunci yang salah untuk gagal eksekusi State Machine dan menghasilkan pesan kesalahan dalam output Response Body.

Langkah 4: Bersihkan

Setelah selesai mencoba API Gateway, Anda dapat meruntuhkan mesin status dan API Gateway menggunakan AWS CDK. Terbitkan `cdk destroy` di direktori utama aplikasi Anda.

AWS Step Functions SDK Ilmu Data untuk Python

AWS Step Functions Data Science SDK adalah perpustakaan sumber terbuka untuk ilmuwan data. Dengan SDK ini, Anda dapat membuat alur kerja yang memproses dan mempublikasikan model pembelajaran mesin menggunakan SageMaker dan Step Functions. Anda juga dapat membuat alur kerja pembelajaran mesin multi-langkah dengan Python yang mengatur AWS infrastruktur dalam skala besar, tanpa harus menyediakan dan mengintegrasikan layanan secara terpisah. AWS

SDK Ilmu Data AWS Step Functions menyediakan API Python yang dapat membuat dan memanggil alur kerja Step Functions. Anda dapat mengelola dan menjalankan alur kerja ini secara langsung dengan Python, serta notebook Jupyter.

Selain membuat alur kerja siap produksi secara langsung dengan Python, AWS Step Functions Data Science SDK memungkinkan Anda menyalin alur kerja itu, bereksperimen dengan opsi baru, dan kemudian memasukkan alur kerja yang disempurnakan ke dalam produksi.

Untuk informasi selengkapnya tentang SDK Ilmu AWS Step Functions Data, lihat berikut ini:

- Proyek di Github
- Dokumentasi SDK

- Contoh notebook berikut, yang tersedia di instance notebook Jupyter di SageMaker konsol dan proyek terkait: GitHub
  - `hello_world_workflow.ipynb`
  - `machine_learning_workflow_abalone.ipynb`
  - `training_pipeline_pytorch_mnist.ipynb`

Menyebarkan mesin status menggunakan Terraform

Terraform by HashiCorp adalah kerangka kerja untuk membangun aplikasi menggunakan infrastruktur sebagai kode (IAc). Dengan Terraform, Anda dapat membuat mesin status dan menggunakan fitur, seperti melihat pratinjau penerapan infrastruktur dan membuat templat yang dapat digunakan kembali. Template Terraform membantu Anda memelihara dan menggunakan kembali kode dengan memecahnya menjadi potongan-potongan yang lebih kecil.

Jika Anda terbiasa dengan Terraform, Anda dapat mengikuti siklus hidup pengembangan yang dijelaskan dalam topik ini sebagai model untuk membuat dan menerapkan mesin status Anda di Terraform. Jika Anda tidak terbiasa dengan Terraform, kami sarankan Anda menyelesaikan lokakarya terlebih dahulu Pengantar Terraform tentang AWS untuk berkenalan dengan Terraform.

> **Tip**
>
> Untuk menerapkan contoh mesin status yang dibuat menggunakan Terraform ke AndaAkun AWS, lihat modul Mengelola mesin status dengan infrastruktur sebagai kode The Workshop. AWS Step Functions

Dalam topik ini:

- Prasyarat
- Siklus hidup pengembangan mesin negara dengan Terraform
- Peran dan kebijakan IAM untuk mesin negara Anda

Prasyarat

Sebelum Anda memulai, pastikan Anda menyelesaikan prasyarat berikut:

- Instal Terraform di mesin Anda. Untuk informasi tentang menginstal Terraform, lihat Menginstal Terraform.
- Instal Step Functions Local di mesin Anda. Kami menyarankan Anda menginstal image Step Functions Local Docker untuk menggunakan Step Functions Local. Untuk informasi selengkapnya, lihat Menguji mesin negara secara lokal.
- Instal AWS SAM CLI. Untuk informasi penginstalan, lihat Menginstal AWS SAM CLI di Panduan AWS Serverless Application ModelPengembang.
- Instal AWS Toolkit for Visual Studio Code untuk melihat diagram alur kerja mesin negara Anda. Untuk informasi penginstalan, lihat Menginstal AWS Toolkit for Visual Studio Code di Panduan AWS Toolkit for Visual Studio Code Pengguna.

Siklus hidup pengembangan mesin negara dengan Terraform

Prosedur berikut menjelaskan bagaimana Anda dapat menggunakan prototipe mesin status yang Anda buat menggunakan Workflow Studio di konsol Step Functions sebagai titik awal untuk pengembangan lokal dengan Terraform dan. AWS Toolkit for Visual Studio Code

Untuk melihat contoh lengkap yang membahas pengembangan mesin status dengan Terraform dan menyajikan praktik terbaik secara detail, lihat Praktik terbaik untuk menulis proyek Step Functions Terraform.

Untuk memulai siklus hidup pengembangan mesin status dengan Terraform

1. Bootstrap proyek Terraform baru dengan perintah berikut.

   ```
   terraform init
   ```

2. Buka konsol Step Functions untuk membuat prototipe untuk mesin status Anda.
3. Di Workflow Studio, lakukan hal berikut:

   a. Buat prototipe alur kerja Anda.

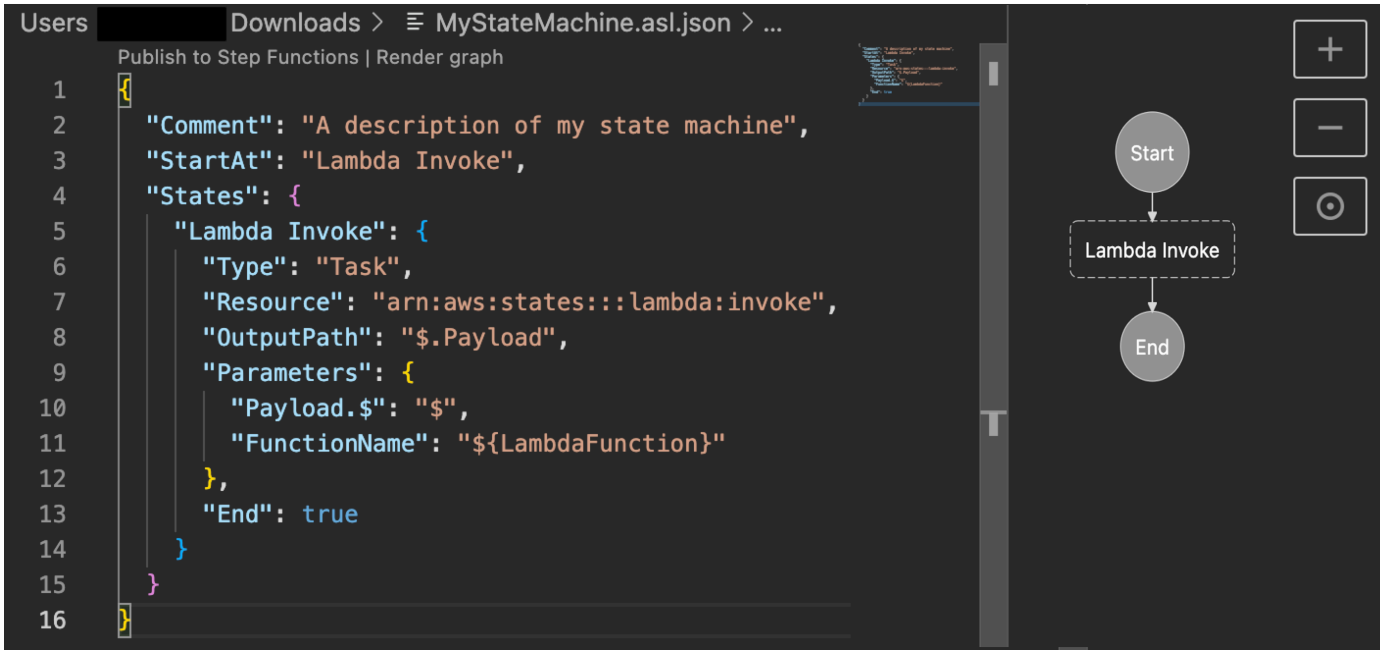
   b. Ekspor definisi Amazon States Language (ASL) dari alur kerja Anda. Untuk melakukan ini, pilih daftar dropdownlist Impor/Ekspor, lalu pilih Ekspor definisi JSON.
4. Simpan definisi ASL yang diekspor dalam direktori proyek Anda.

   Anda meneruskan definisi ASL yang diekspor sebagai parameter input ke sumber daya `aws_sfn_state_machine`Terraform yang menggunakan fungsi tersebut. `templatefile`

Fungsi ini digunakan di dalam bidang definisi yang melewati definisi ASL yang diekspor dan substitusi variabel apa pun.

> **Tip**
>
> Karena file definisi ASL dapat berisi blok teks yang panjang, kami sarankan Anda menghindari metode EOF sebaris. Ini membuatnya lebih mudah untuk mengganti parameter ke dalam definisi mesin status Anda.

5. (Opsional) Perbarui definisi ASL dalam IDE Anda dan visualisasikan perubahan Anda menggunakan. AWS Toolkit for Visual Studio Code



Untuk menghindari terus-menerus mengekspor definisi Anda dan memfaktorkannya kembali ke proyek Anda, kami sarankan Anda membuat pembaruan secara lokal di IDE Anda dan melacak pembaruan ini dengan Git.

6. Uji alur kerja Anda menggunakan Step Functions Local.

> **Tip**
>
> Anda juga dapat menguji integrasi layanan secara lokal dengan fungsi Lambda dan API Gateway API di mesin status Anda menggunakan CLI Local. AWS SAM

7. Pratinjau mesin status Anda dan AWS sumber daya lainnya sebelum menerapkan mesin status. Untuk melakukan ini, jalankan perintah berikut.

```
terraform plan
```

8. Terapkan mesin status Anda dari lingkungan lokal Anda atau melalui pipa CI/CD menggunakan perintah berikut.

```
terraform apply
```

9. (Opsional) Bersihkan sumber daya Anda dan hapus mesin status menggunakan perintah berikut.

```
terraform destroy
```

Peran dan kebijakan IAM untuk mesin negara Anda

Gunakan kebijakan integrasi layanan Terraform untuk menambahkan izin IAM yang diperlukan ke mesin status Anda, misalnya, izin untuk menjalankan fungsi Lambda. Anda juga dapat menentukan peran dan kebijakan eksplisit dan mengaitkannya dengan mesin status Anda.

Contoh kebijakan IAM berikut memberikan akses mesin status Anda untuk memanggil fungsi Lambda bernama. *myFunction*

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunction"
      ],
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:myFunction"
    }
  ]
}
```

Kami juga merekomendasikan penggunaan sumber `aws_iam_policy_document`data saat mendefinisikan kebijakan IAM untuk mesin negara Anda di Terraform. Ini membantu Anda memeriksa apakah kebijakan Anda salah bentuk dan mengganti sumber daya apa pun dengan variabel.

Contoh kebijakan IAM berikut menggunakan sumber `aws_iam_policy_document` data dan memberikan akses mesin status Anda untuk memanggil fungsi Lambda bernama. *myFunction*

```
data "aws_iam_policy_document" "state_machine_role_policy" {

  statement {
    effect = "Allow"

    actions = [
      "lambda:InvokeFunction"
    ]

    resources = ["${aws_lambda_function.[[myFunction]].arn}:*"]
  }

}
```

> **Tip**
>
> Untuk melihat lebih banyak pola AWS arsitektur lanjutan yang diterapkan dengan Terraform, lihat contoh Terraform di Koleksi Alur Kerja Tanah Tanpa Server.

Pengujian dan Debugging

Step Functionsmenyediakan berbagai cara untuk menguji dan men-debug mesin status. Misalnya, Anda dapat menguji dan men-debug mesin status di konsol, menggunakan TestStateAPI untuk menguji status individual, atau menggunakan Step Functions Local untuk menguji mesin status secara lokal.

Menggunakan TestStateAPI, Anda memberikan definisi status tunggal dan menjalankannya. Anda dapat menguji satu status tanpa membuat mesin status atau memperbarui mesin status yang ada.

Step FunctionsLokal adalah versi download Step Functions yang memungkinkan Anda mengembangkan dan menguji aplikasi menggunakan versi Step Functions berjalan di lingkungan pengembangan Anda sendiri. Dengan Step Functions Local, Anda dapat menjalankan mesin status untuk menguji aliran data input dan outputnya, integrasi dengan layanan yang didukung, dan lainnya di lingkungan pengembangan lokal Anda.

Topik

- Menggunakan TestState API untuk menguji status
- Menguji mesin negara secara lokal

Menggunakan TestState API untuk menguji status

TestStateAPI menerima definisi status tunggal dan menjalankannya. Anda dapat menguji status tanpa membuat mesin status atau memperbarui mesin status yang ada.

Dengan menggunakan TestState API, Anda dapat menguji hal-hal berikut:

- Aliran data pemrosesan input dan output suatu negara.
- Layanan AWS Integrasi dengan Layanan AWS permintaan dan tanggapan lainnya
- Permintaan dan respons HTTP Task

Untuk menguji status, Anda juga dapat menggunakan Step Functionskonsol, AWS Command Line Interface (AWS CLI), atau SDK.

`TestState`API mengasumsikan peran IAM yang harus berisi IAM izin yang diperlukan untuk sumber daya yang diakses status Anda. Untuk informasi tentang izin yang mungkin diperlukan oleh negara bagian, lihatIAMizin untuk menggunakan API TestState .

Topik

- Pertimbangan tentang penggunaan API TestState
- Menggunakan level inspeksi di TestState API
- IAMizin untuk menggunakan API TestState
- Menguji status (Konsol)
- Menguji status menggunakan AWS CLI
- Menguji dan men-debug aliran data input dan output

Pertimbangan tentang penggunaan API TestState

Menggunakan TestStateAPI, Anda dapat menguji hanya satu status pada satu waktu. Status yang dapat Anda uji meliputi yang berikut:

- Semua jenis Tugas, kecuali Aktivitas
- Diteruskan
- Tunggu
- Pilihan
- Berhasil
- Gagal

Saat menggunakan `TestState` API, ingatlah pertimbangan berikut.

- TestState API tidak menyertakan dukungan untuk hal-hal berikut:
  - Tugasmenyatakan yang menggunakan jenis sumber daya berikut:
    - Aktivitas
    - Pola integrasi layanan dari jenis `.sync` atau `.waitForTaskToken`
  - Paralelnegara
  - Mapnegara
- Tes dapat berjalan hingga lima menit. Jika tes melebihi durasi ini, itu gagal dengan `States.Timeout` kesalahan.

Menggunakan level inspeksi di TestState API

Untuk menguji status menggunakan TestStateAPI, Anda memberikan definisi status tersebut. Tes kemudian mengembalikan output. Untuk setiap negara bagian, Anda dapat menentukan jumlah detail yang ingin Anda lihat dalam hasil tes. Detail ini memberikan informasi tambahan tentang status yang Anda uji. Misalnya, jika Anda telah menggunakan filter pemrosesan data input dan output, seperti `InputPath`atau `ResultPath`dalam keadaan, Anda dapat melihat hasil pemrosesan data antara dan akhir.

Step Functionsmenyediakan level berikut untuk menentukan detail yang ingin Anda lihat:

- INFO
- DEBUG
- JEJAK

Semua level ini juga mengembalikan `status` dan `nextState` bidang. `status`menunjukkan status eksekusi negara. Misalnya,, `SUCCEEDED`dan`FAILED`,`RETRIABLE`, dan`CAUGHT_ERROR`. `nextState`menunjukkan nama negara bagian berikutnya untuk transisi ke. Jika Anda belum menentukan status berikutnya dalam definisi Anda, bidang ini mengembalikan nilai kosong.

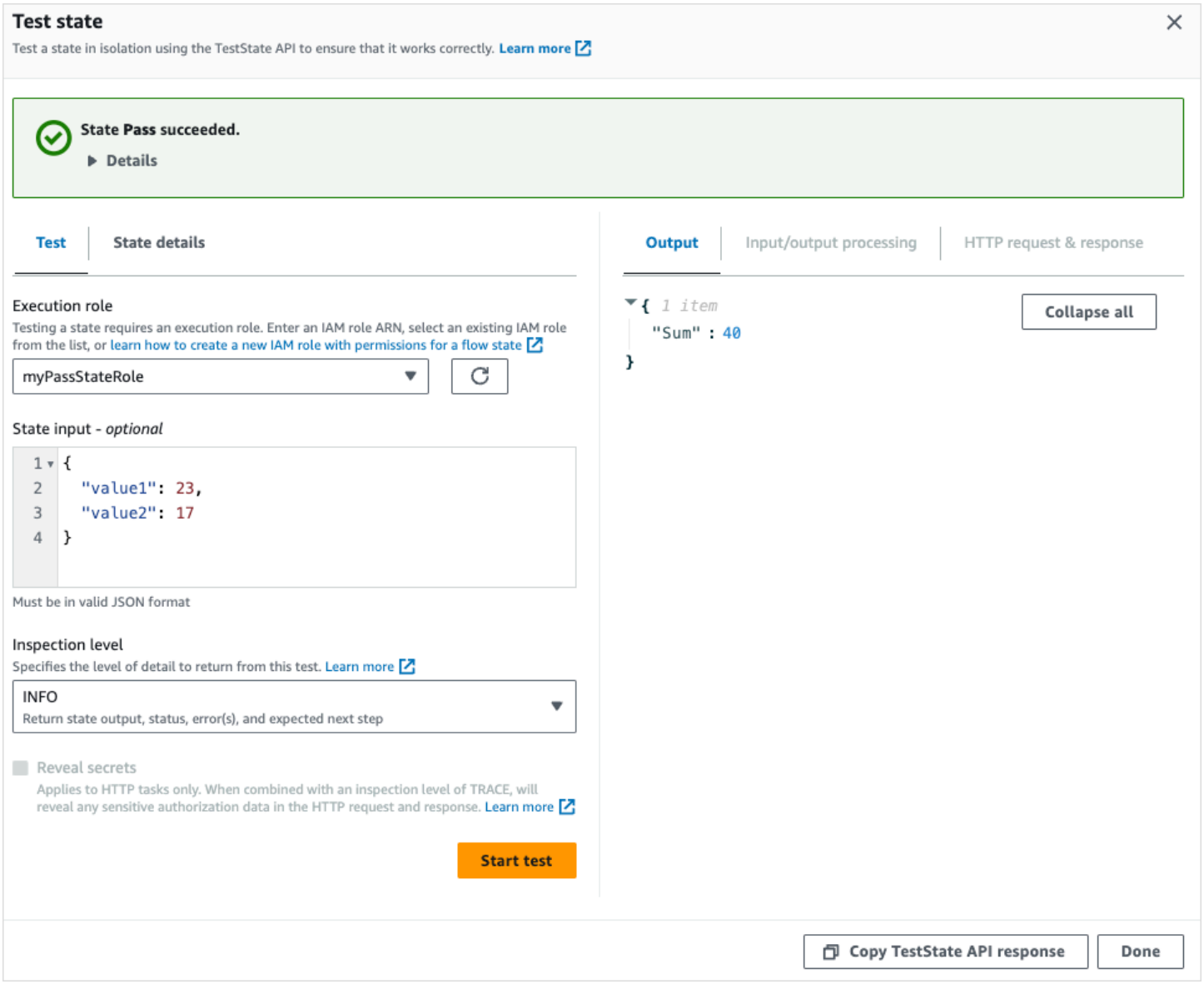
Untuk informasi tentang pengujian status menggunakan tingkat inspeksi ini di Step Functions konsol danAWS CLI, lihat Menguji status (Konsol) danMenguji status menggunakan AWS CLI.

INFO InspectionLevel

Jika tes berhasil, level ini menunjukkan output status. Jika tes gagal, level ini menunjukkan output kesalahan. Secara default, Step Functions tetapkan level Inspeksi ke INFO jika Anda tidak menentukan level.

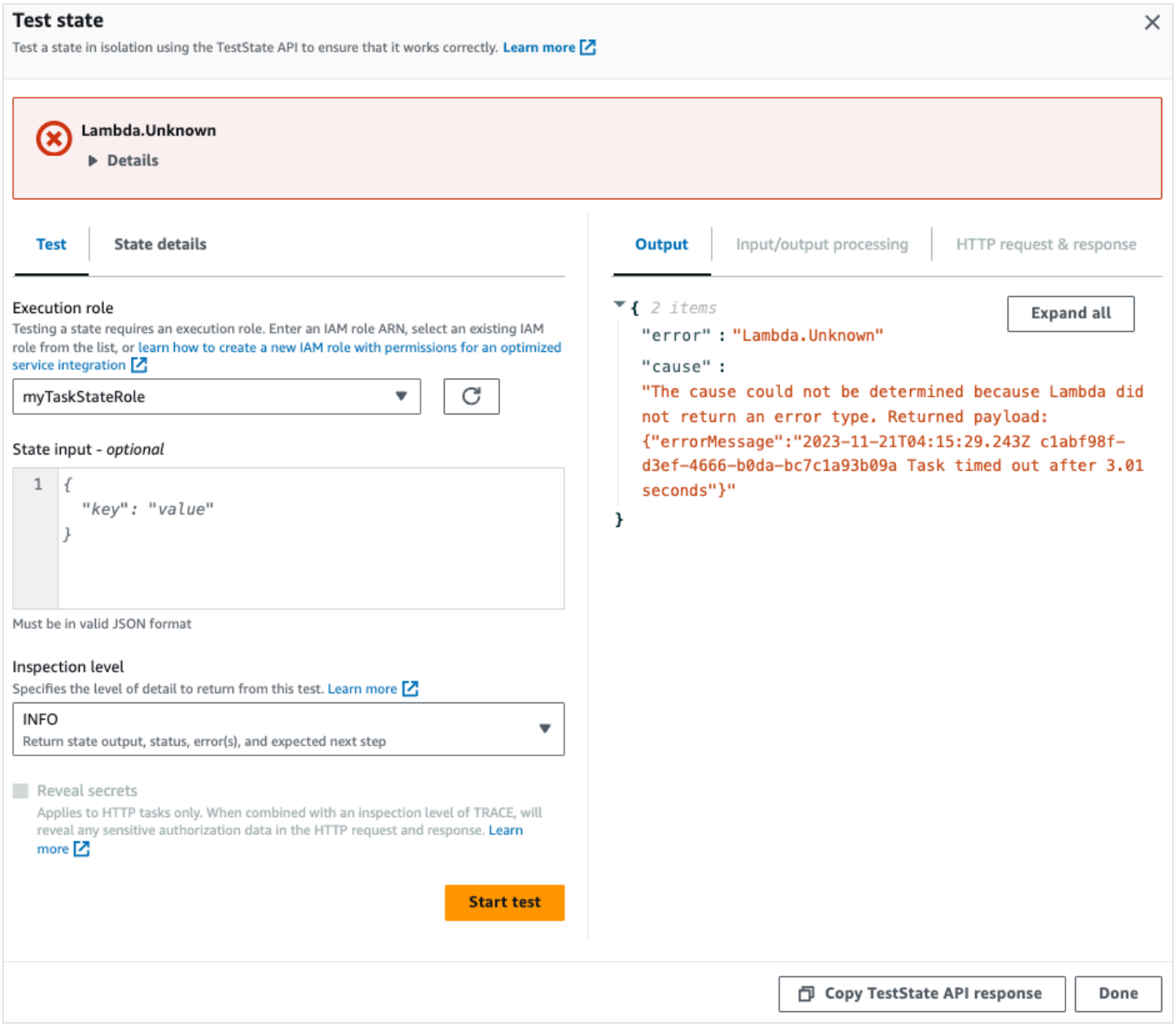
Contoh tes dengan level INFO yang berhasil

Gambar berikut menunjukkan tes untuk status Lulus yang berhasil. Tingkat Inspeksi untuk status ini diatur ke INFO dan output untuk status muncul di tab Output.



Contoh pengujian dengan level INFO yang gagal

Gambar berikut menunjukkan pengujian yang gagal untuk status Tugas saat tingkat Inspeksi disetel ke INFO. Tab Output menunjukkan output kesalahan yang mencakup nama kesalahan dan penjelasan rinci tentang penyebab kesalahan itu.



DEBUG Inspeksi Level

Jika tes berhasil, level ini menunjukkan output status dan hasil pemrosesan data input dan output.

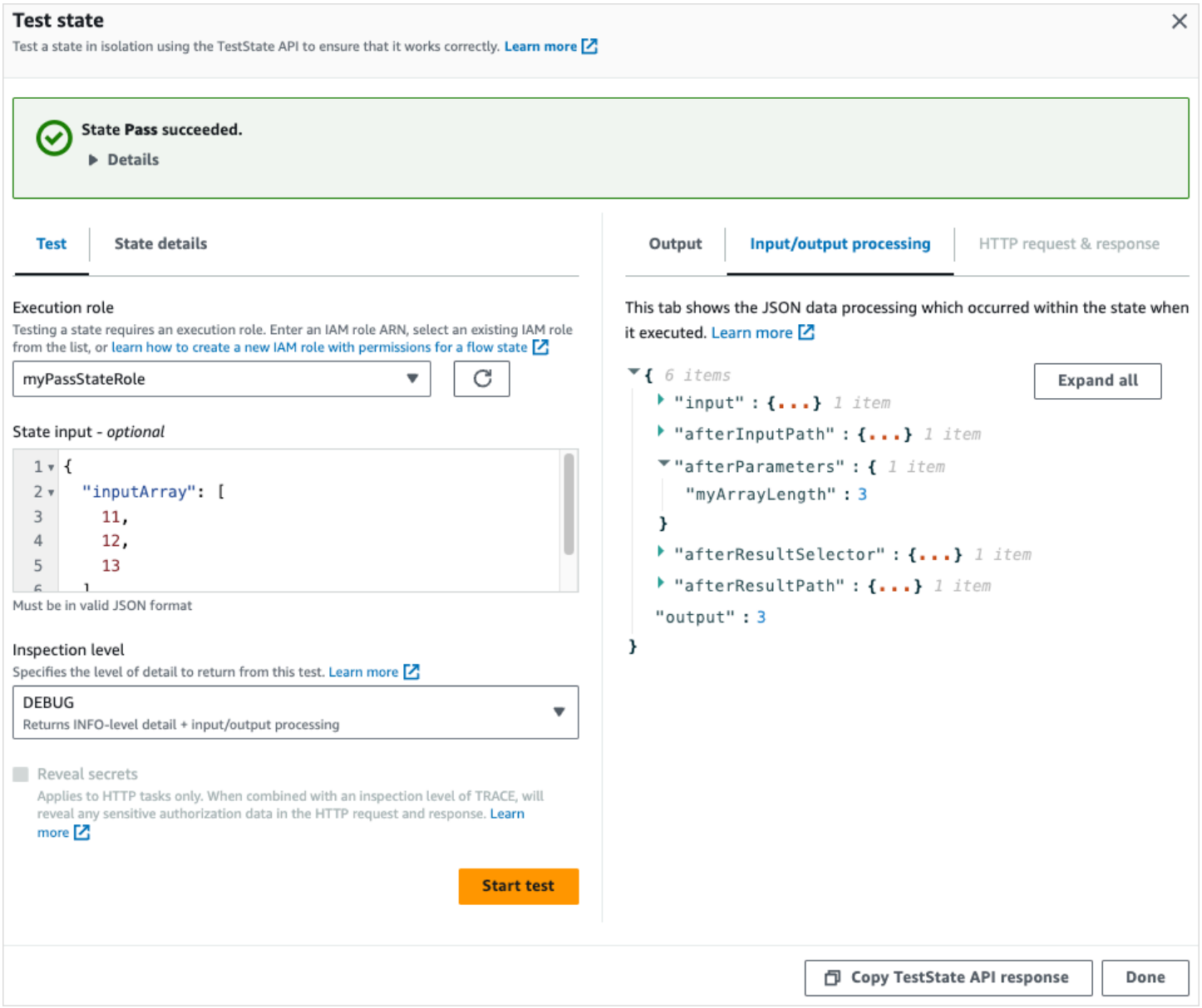
Jika tes gagal, level ini menunjukkan output kesalahan. Tingkat ini menunjukkan hasil pemrosesan data menengah hingga titik kegagalan. Misalnya, katakan bahwa Anda menguji status Tugas yang memanggil Lambda fungsi. Bayangkan Anda telah menerapkanInputPath,Parameter,ResultPath, dan OutputPath filter ke status Tugas. Katakan bahwa doa gagal. Dalam hal ini, DEBUG level menunjukkan hasil pemrosesan data berdasarkan penerapan filter dalam urutan sebagai berikut:

- `input`— Masukan status mentah
- `afterInputPath`— Masukan setelah Step Functions menerapkan `InputPath` filter.
- `afterParameters`- Input efektif setelah Step Functions menerapkan `Parameters` filter.

Informasi diagnostik yang tersedia di level ini dapat membantu Anda memecahkan masalah yang terkait dengan integrasi layanan atau aliran pemrosesan data input dan output yang mungkin telah Anda tentukan.

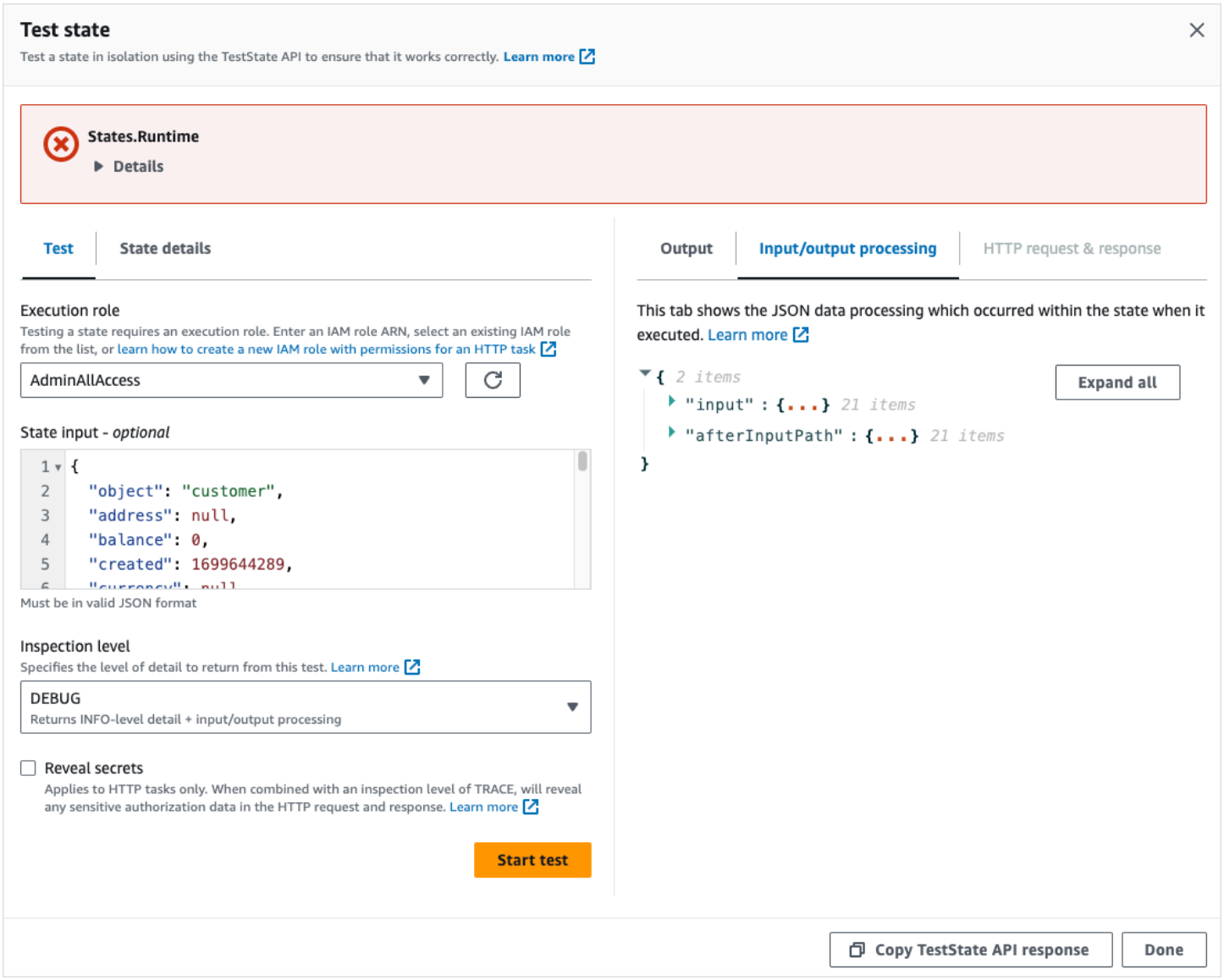
Contoh pengujian dengan level DEBUG yang berhasil

Gambar berikut menunjukkan tes untuk status Lulus yang berhasil. Tingkat Inspeksi untuk status ini diatur ke DEBUG. Tab pemrosesan input/output pada gambar berikut menunjukkan hasil penerapan `Parameters`pada input yang disediakan untuk keadaan ini.



Contoh pengujian dengan level DEBUG yang gagal

Gambar berikut menunjukkan pengujian yang gagal untuk status Tugas saat tingkat Inspeksi disetel ke DEBUG. Tab pemrosesan input/output pada gambar berikut menunjukkan hasil pemrosesan data input dan output untuk status hingga titik kegagalannya.



TRACE Inspection Level

Step Functionsmenyediakan tingkat TRACE untuk menguji Tugas HTTP. Level ini mengembalikan informasi tentang permintaan HTTP yang Step Functions membuat dan merespons yang dikembalikan oleh API pihak ketiga. Respons mungkin berisi informasi, seperti header dan badan permintaan. Selain itu, Anda dapat melihat output status dan hasil pemrosesan data input dan output di level ini.

Jika tes gagal, level ini menunjukkan output kesalahan.

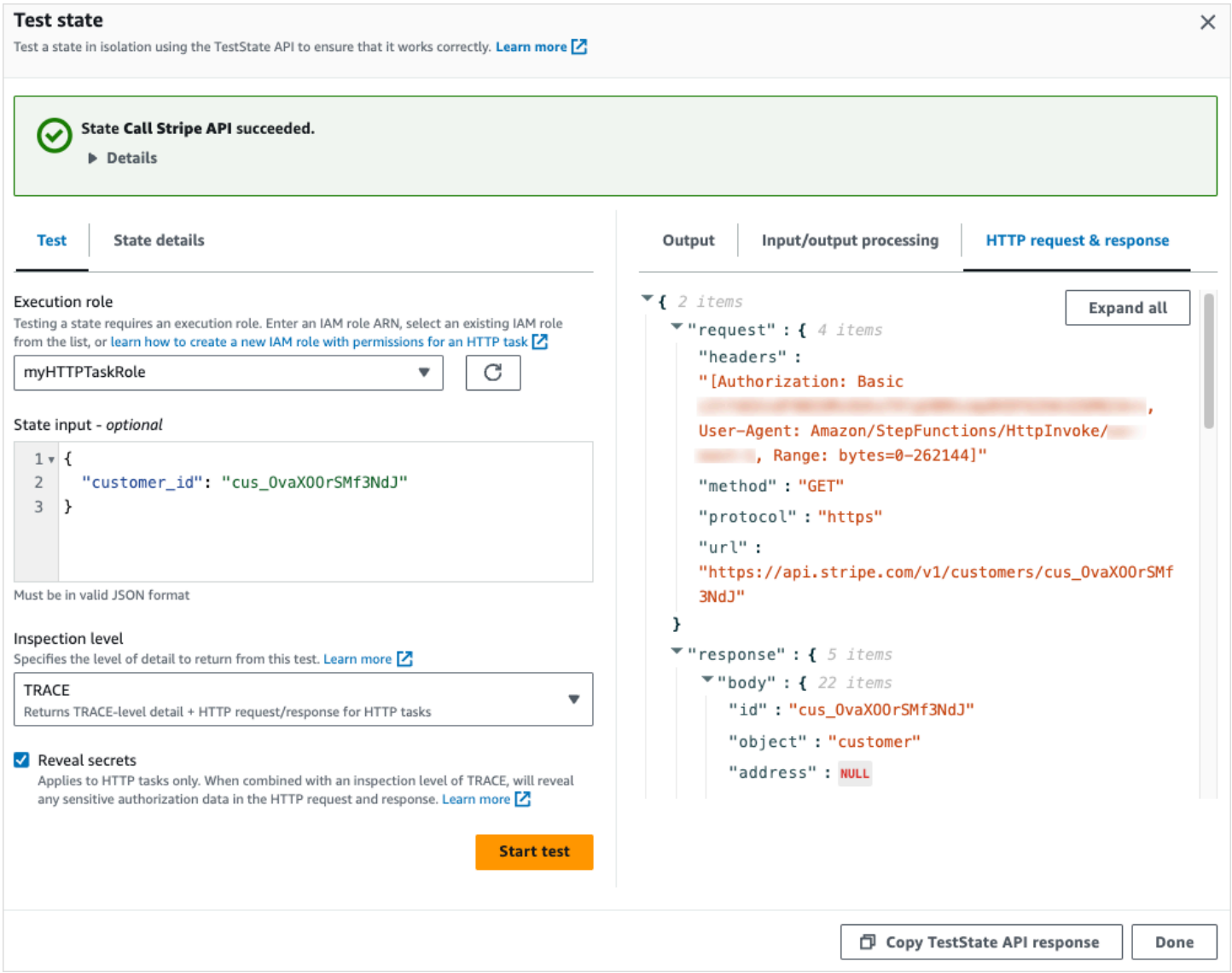
Level ini hanya berlaku untuk HTTP Task. Step Functionsmelempar kesalahan jika Anda menggunakan level ini untuk tipe status lainnya.

Saat Anda mengatur level Inspeksi ke TRACE, Anda juga dapat melihat rahasia yang disertakan dalam EventBridge koneksi. Untuk melakukan ini, Anda harus mengatur `revealSecrets` parameter ke `true` dalam TestStateAPI. Selain itu, Anda harus memastikan bahwa IAM pengguna yang memanggil TestState API memiliki izin untuk `states:RevealSecrets` tindakan tersebut. Untuk contoh IAM kebijakan yang menetapkan `states:RevealSecrets` izin, lihatIAMizin untuk menggunakan API TestState . Tanpa izin ini, Step Functions melempar kesalahan akses ditolak.

Jika Anda mengatur `revealSecrets` parameter ke`false`, Step Functions menghilangkan semua rahasia dalam permintaan HTTP dan data respons.

Contoh tes dengan level TRACE yang berhasil

Gambar berikut menunjukkan tes untuk Tugas HTTP yang berhasil. Tingkat Inspeksi untuk status ini diatur ke TRACE. Tab permintaan & respons HTTP pada gambar berikut menunjukkan hasil panggilan API pihak ketiga.



IAMizin untuk menggunakan API TestState

IAMPengguna yang memanggil `TestState` API harus memiliki izin untuk melakukan `states:TestState` dan `iam:PassRole` tindakan. Selain itu, jika Anda menyetel parameter RevealSecrets `ketrue`, Anda harus memastikan bahwa IAM pengguna memiliki izin untuk melakukan tindakan. `states:RevealSecrets` Tanpa izin ini, Step Functions melempar kesalahan akses ditolak.

Anda juga harus memastikan bahwa peran eksekusi Anda berisi IAM izin yang diperlukan untuk sumber daya yang diakses negara Anda. Untuk informasi tentang izin yang mungkin diperlukan oleh negara bagian, lihat Mengelola peran eksekusi.

Contoh IAM kebijakan berikut menetapkan `states:RevealSecrets` izin`states:TestState`,`iam:PassRole`, dan.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:TestState",
        "states:RevealSecrets",
        "iam:PassRole"
      ],
      "Resource": "*"
    }
  ]
}
```

Menguji status (Konsol)

Anda dapat menguji status di konsol dan memeriksa output status atau aliran pemrosesan data input dan output. Untuk Tugas HTTP, Anda dapat menguji permintaan dan respons HTTP mentah.

Untuk menguji suatu negara

1. Buka Konsol Step Functions.
2. Pilih Create state machine untuk mulai membuat state machine atau pilih state machine yang sudah ada.
3. Di Mode desain Workflow Studio, pilih status yang ingin Anda uji.
4. Pilih status Uji di Inspector panel Workflow Studio.
5. Dalam kotak dialog Test state, lakukan hal berikut:

    a. Untuk peran Eksekusi, pilih peran eksekusi untuk menguji status. Pastikan Anda memiliki IAMizin yang diperlukan untuk status yang ingin Anda uji.

    b. (Opsional) Berikan masukan JSON apa pun yang dibutuhkan status yang Anda pilih untuk pengujian.

    c. Untuk tingkat Inspeksi, pilih salah satu opsi berikut berdasarkan nilai yang ingin Anda lihat:

- INFO - Menampilkan output status di tab Output jika tes berhasil. Jika tes gagal, INFO menunjukkan output kesalahan, yang mencakup nama kesalahan dan penjelasan rinci tentang penyebab kesalahan itu. Secara default, Step Functions tetapkan level Inspeksi ke INFO jika Anda tidak memilih level.
- DEBUG - Menunjukkan output status dan hasil pemrosesan data input dan output jika pengujian berhasil. Jika pengujian gagal, DEBUG menunjukkan output kesalahan, yang mencakup nama kesalahan dan penjelasan rinci tentang penyebab kesalahan itu.
- TRACE - Menampilkan permintaan dan respons HTTP mentah, dan berguna untuk memverifikasi header, parameter kueri, dan detail spesifik API lainnya. Opsi ini hanya tersedia untuk Tugas HTTP.

  Secara opsional, Anda dapat memilih untuk memilih Mengungkapkan rahasia. Dalam kombinasi dengan TRACE, pengaturan ini memungkinkan Anda melihat data sensitif yang disisipkan EventBridge koneksi, seperti kunci API. Identitas IAM pengguna yang Anda gunakan untuk mengakses konsol harus memiliki izin untuk melakukan `states:RevealSecrets` tindakan. Tanpa izin ini, Step Functions melempar kesalahan akses ditolak saat Anda memulai pengujian. Untuk contoh IAM kebijakan yang menetapkan `states:RevealSecrets` izin, lihatIAMizin untuk menggunakan API TestState .

d. Pilih Mulai tes.

Menguji status menggunakan AWS CLI

Anda dapat menguji status yang didukung menggunakan TestStateAPI diAWS CLI. API ini menerima definisi status dan menjalankannya.

Untuk setiap negara bagian, Anda dapat menentukan jumlah detail yang ingin Anda lihat dalam hasil tes. Rincian ini memberikan informasi tambahan tentang eksekusi negara, termasuk input dan output hasil pengolahan data dan permintaan HTTP dan informasi respon. Contoh berikut menampilkan berbagai level inspeksi yang dapat Anda tentukan untuk TestState API. Ingatlah untuk mengganti teks yang *`dicetak miring dengan informasi`* spesifik sumber daya Anda.

Bagian ini berisi contoh-contoh berikut yang menjelaskan bagaimana Anda dapat menggunakan berbagai tingkat inspeksi yang Step Functions disediakan dalamAWS CLI:

- Menggunakan INFO InspectionLevel

- Menggunakan DEBUG InspectionLevel
- Menggunakan TRACE InspectionLevel
- Menggunakan utilitas jq AWS CLI untuk memfilter dan mencetak respons HTTP yang dikembalikan TestState API

Contoh 1: Menggunakan INFO InspectionLevel untuk menguji status Choice

Untuk menguji status menggunakan `INFO` InspectionLevel di AWS CLI, jalankan `test-state` perintah seperti yang ditunjukkan pada contoh berikut.

```
aws stepfunctions test-state \
    --definition '{"Type": "Choice", "Choices": [{"Variable": "$.number",
 "NumericEquals": 1, "Next": "Equals 1"}, {"Variable": "$.number", "NumericEquals": 2,
 "Next": "Equals 2"}], "Default": "No Match"}' \
    --role-arn arn:aws:iam::123456789012:role/myRole \
    --input '{"number": 2}'
```

Contoh ini menggunakan status Pilihan untuk menentukan jalur eksekusi untuk status berdasarkan input numerik yang Anda berikan. Secara default, Step Functions atur `inspectionLevel` ke `INFO` jika Anda tidak menetapkan level.

Step Functionsmengembalikan output berikut.

```
{
    "output": "{\"number\": 2}",
    "nextState": "Equals 2",
    "status": "SUCCEEDED"
}
```

Contoh 2: Menggunakan DEBUG InspectionLevel untuk men-debug pemrosesan data input dan output dalam status Pass

Untuk menguji status menggunakan `DEBUG` InspectionLevel di AWS CLI, jalankan `test-state` perintah seperti yang ditunjukkan pada contoh berikut.

```
aws stepfunctions test-state \
    --definition '{"Type": "Pass", "InputPath": "$.payload", "Parameters": {"data": 1},
 "ResultPath": "$.result", "OutputPath": "$.result.data", "Next": "Another State"}' \
    --role-arn arn:aws:iam::123456789012:role/myRole \
    --input '{"payload": {"foo": "bar"}}' \
```

```
    --inspection-level DEBUG
```

Contoh ini menggunakan Diteruskan status untuk menampilkan bagaimana Step Functions filter dan memanipulasi input data JSON menggunakan input dan output filter pengolahan data. Contoh ini menggunakan filter ini:`InputPath`,`Parameter`,`ResultPath`, dan`OutputPath`.

Step Functionsmengembalikan output berikut.

```
{
    "output": "1",
    "inspectionData": {
        "input": "{\"payload\": {\"foo\": \"bar\"}}",
        "afterInputPath": "{\"foo\":\"bar\"}",
        "afterParameters": "{\"data\":1}",
        "afterResultSelector": "{\"data\":1}",
        "afterResultPath": "{\"payload\":{\"foo\":\"bar\"},\"result\":{\"data\":1}}"
    },
    "nextState": "Another State",
    "status": "SUCCEEDED"
}
```

Contoh 3: Menggunakan TRACE InspectionLevel dan RevealSecrets untuk memeriksa permintaan HTTP yang dikirim ke API pihak ketiga

Untuk menguji Tugas HTTP menggunakan `TRACE` InspectionLevel bersama dengan parameter RevealSecrets di AWS CLI, jalankan `test-state` perintah seperti yang ditunjukkan pada contoh berikut.

```
aws stepfunctions test-state \
    --definition '{"Type": "Task", "Resource": "arn:aws:states:::http:invoke",
 "Parameters": {"Method": "GET", "Authentication": {"ConnectionArn":
 "arn:aws:events:us-
east-1:123456789012:connection/MyConnection/0000000-0000-0000-0000-000000000000"},
 "ApiEndpoint": "https://httpbin.org/get", "Headers": {"definitionHeader": "h1"},
 "RequestBody": {"message": "Hello from Step Functions!"}, "QueryParameters":
 {"queryParam": "q1"}}, "End": true}' \
    --role-arn arn:aws:iam::123456789012:role/myRole \
    --inspection-level TRACE \
    --reveal-secrets
```

Contoh ini menguji apakah Tugas HTTP memanggil API pihak ketiga yang ditentukan`https://httpbin.org/`. Ini juga menunjukkan permintaan HTTP dan data respons untuk panggilan API.


```
{
    "output": "{\"Headers\":{\"date\":[\"Tue, 21 Nov 2023 00:06:17 GMT\"],
\"access-control-allow-origin\":[\"*\"],\"content-length\":[\"620\"],\"server\":
[\"gunicorn/19.9.0\"],\"access-control-allow-credentials\":[\"true\"],\"content-
type\":[\"application/json\"]},\"ResponseBody\":{\"args\":{\"QueryParam1\":
\"QueryParamValue1\",\"queryParam\":\"q1\"},\"headers\":{\"Authorization
\":\"Basic XXXXXXXX\",\"Content-Type\":\"application/json; charset=UTF-8\",
\"Customheader1\":\"CustomHeaderValue1\",\"Definitionheader\":\"h1\",\"Host\":
\"httpbin.org\",\"Range\":\"bytes=0-262144\",\"Transfer-Encoding\":\"chunked\",
\"User-Agent\":\"Amazon|StepFunctions|HttpInvoke|us-east-1\",\"X-Amzn-Trace-Id\":
\"Root=1-0000000-0000-0000-0000-000000000000\"},\"origin\":\"12.34.567.891\",\"url\":
\"https://httpbin.org/get?queryParam=q1&QueryParam1=QueryParamValue1\"},\"StatusCode
\":200,\"StatusText\":\"OK\"}",
    "inspectionData": {
        "input": "{}",
        "afterInputPath": "{}",
        "afterParameters": "{\"Method\":\"GET\",\"Authentication\":{\"ConnectionArn
\":\"arn:aws:events:us-east-1:123456789012:connection/foo/a59c10f0-a315-4c1f-
be6a-559b9a0c6250\"},\"ApiEndpoint\":\"https://httpbin.org/get\",\"Headers\":
{\"definitionHeader\":\"h1\"},\"RequestBody\":{\"message\":\"Hello from Step Functions!
\"},\"QueryParameters\":{\"queryParam\":\"q1\"}}",
        "result": "{\"Headers\":{\"date\":[\"Tue, 21 Nov 2023 00:06:17 GMT\"],
\"access-control-allow-origin\":[\"*\"],\"content-length\":[\"620\"],\"server\":
[\"gunicorn/19.9.0\"],\"access-control-allow-credentials\":[\"true\"],\"content-
type\":[\"application/json\"]},\"ResponseBody\":{\"args\":{\"QueryParam1\":
\"QueryParamValue1\",\"queryParam\":\"q1\"},\"headers\":{\"Authorization
\":\"Basic XXXXXXXX\",\"Content-Type\":\"application/json; charset=UTF-8\",
\"Customheader1\":\"CustomHeaderValue1\",\"Definitionheader\":\"h1\",\"Host\":
\"httpbin.org\",\"Range\":\"bytes=0-262144\",\"Transfer-Encoding\":\"chunked\",
\"User-Agent\":\"Amazon|StepFunctions|HttpInvoke|us-east-1\",\"X-Amzn-Trace-Id\":
\"Root=1-0000000-0000-0000-0000-000000000000\"},\"origin\":\"12.34.567.891\",\"url\":
\"https://httpbin.org/get?queryParam=q1&QueryParam1=QueryParamValue1\"},\"StatusCode
\":200,\"StatusText\":\"OK\"}",
        "afterResultSelector": "{\"Headers\":{\"date\":[\"Tue, 21 Nov 2023
 00:06:17 GMT\"],\"access-control-allow-origin\":[\"*\"],\"content-length\":
[\"620\"],\"server\":[\"gunicorn/19.9.0\"],\"access-control-allow-credentials
\":[\"true\"],\"content-type\":[\"application/json\"]},\"ResponseBody\":{\"args
\":{\"QueryParam1\":\"QueryParamValue1\",\"queryParam\":\"q1\"},\"headers\":
{\"Authorization\":\"Basic XXXXXXXX\",\"Content-Type\":\"application/json;
 charset=UTF-8\",\"Customheader1\":\"CustomHeaderValue1\",\"Definitionheader\":\"h1\",
```

\"Host\":\"httpbin.org\",\"Range\":\"bytes=0-262144\",\"Transfer-Encoding\":\"chunked
\",\"User-Agent\":\"Amazon|StepFunctions|HttpInvoke|us-east-1\",\"X-Amzn-Trace-Id\":
\"Root=1-0000000-0000-0000-0000-000000000000\"},\"origin\":\"12.34.567.891\",\"url\":
\"https://httpbin.org/get?queryParam=q1&QueryParam1=QueryParamValue1\"},\"StatusCode
\":200,\"StatusText\":\"OK\"}",
        "afterResultPath": "{\"Headers\":{\"date\":[\"Tue, 21 Nov 2023 00:06:17
 GMT\"],\"access-control-allow-origin\":[\"*\"],\"content-length\":[\"620\"],
\"server\":[\"gunicorn/19.9.0\"],\"access-control-allow-credentials\":[\"true\"],
\"content-type\":[\"application/json\"]},\"ResponseBody\":{\"args\":{\"QueryParam1\":
\"QueryParamValue1\",\"queryParam\":\"q1\"},\"headers\":{\"Authorization\":
\"Basic XXXXXXXX\",\"Content-Type\":\"application/json; charset=UTF-8\",
\"Customheader1\":\"CustomHeaderValue1\",\"Definitionheader\":\"h1\",\"Host\":
\"httpbin.org\",\"Range\":\"bytes=0-262144\",\"Transfer-Encoding\":\"chunked\",
\"User-Agent\":\"Amazon|StepFunctions|HttpInvoke|us-east-1\",\"X-Amzn-Trace-Id\":
\"Root=1-0000000-0000-0000-0000-000000000000\"},\"origin\":\"12.34.567.891\",\"url\":
\"https://httpbin.org/get?queryParam=q1&QueryParam1=QueryParamValue1\"},\"StatusCode
\":200,\"StatusText\":\"OK\"}",
        "request": {
            "protocol": "https",
            "method": "GET",
            "url": "https://httpbin.org/get?
queryParam=q1&QueryParam1=QueryParamValue1",
            "headers": "[definitionHeader: h1, Authorization: Basic XXXXXXXX,
 CustomHeader1: CustomHeaderValue1, User-Agent: Amazon|StepFunctions|HttpInvoke|us-
east-1, Range: bytes=0-262144]",
            "body": "{\"message\":\"Hello from Step Functions!\",\"BodyKey1\":
\"BodyValue1\"}"
        },
        "response": {
            "protocol": "https",
            "statusCode": "200",
            "statusMessage": "OK",
            "headers": "[date: Tue, 21 Nov 2023 00:06:17 GMT, content-type:
 application/json, content-length: 620, server: gunicorn/19.9.0, access-control-allow-
origin: *, access-control-allow-credentials: true]",
            "body": "{\n \"args\": {\n   \"QueryParam1\": \"QueryParamValue1\", \n
    \"queryParam\": \"q1\"\n }, \n \"headers\": {\n   \"Authorization\": \"Basic
 XXXXXXXX\", \n   \"Content-Type\": \"application/json; charset=UTF-8\", \n
 \"Customheader1\": \"CustomHeaderValue1\", \n   \"Definitionheader\": \"h1\", \n
   \"Host\": \"httpbin.org\", \n   \"Range\": \"bytes=0-262144\", \n   \"Transfer-
Encoding\": \"chunked\", \n   \"User-Agent\": \"Amazon|StepFunctions|HttpInvoke|us-
east-1\", \n   \"X-Amzn-Trace-Id\": \"Root=1-0000000-0000-0000-0000-000000000000\"\n
  }, \n \"origin\": \"12.34.567.891\", \n \"url\": \"https://httpbin.org/get?
queryParam=q1&QueryParam1=QueryParamValue1\"\n}\n"

```
        }
    },
    "status": "SUCCEEDED"
}
```

Contoh 4: Menggunakan utilitas jq untuk memfilter dan mencetak respons yang dikembalikan TestState API

TestState API mengembalikan data JSON sebagai string yang lolos dalam responsnya. AWS CLIContoh berikut memperluas Contoh 3 dan menggunakan `jq` utilitas untuk memfilter dan mencetak respons HTTP yang dikembalikan TestState API dalam format yang dapat dibaca manusia. Untuk informasi tentang `jq` dan petunjuk pemasangannya, lihat jq on GitHub.

```
aws stepfunctions test-state \
    --definition '{"Type": "Task", "Resource": "arn:aws:states:::http:invoke",
 "Parameters": {"Method": "GET", "Authentication": {"ConnectionArn":
 "arn:aws:events:us-
east-1:123456789012:connection/MyConnection/0000000-0000-0000-0000-000000000000"},
 "ApiEndpoint": "https://httpbin.org/get", "Headers": {"definitionHeader": "h1"},
 "RequestBody": {"message": "Hello from Step Functions!"}, "QueryParameters":
 {"queryParam": "q1"}}, "End": true}' \
    --role-arn arn:aws:iam::123456789012:role/myRole \
    --inspection-level TRACE \
    --reveal-secrets \
    | jq '.inspectionData.response.body | fromjson'
```

Contoh berikut menunjukkan output yang dikembalikan dalam format yang dapat dibaca manusia.

```
{
  "args": {
    "QueryParam1": "QueryParamValue1",
    "queryParam": "q1"
  },
  "headers": {
    "Authorization": "Basic XXXXXXXX",
    "Content-Type": "application/json; charset=UTF-8",
    "Customheader1": "CustomHeaderValue1",
    "Definitionheader": "h1",
    "Host": "httpbin.org",
    "Range": "bytes=0-262144",
    "Transfer-Encoding": "chunked",
```

```
    "User-Agent": "Amazon|StepFunctions|HttpInvoke|us-east-1",
    "X-Amzn-Trace-Id": "Root=1-0000000-0000-0000-0000-000000000000"
  },
  "origin": "12.34.567.891",
  "url": "https://httpbin.org/get?queryParam=q1&QueryParam1=QueryParamValue1"
}
```

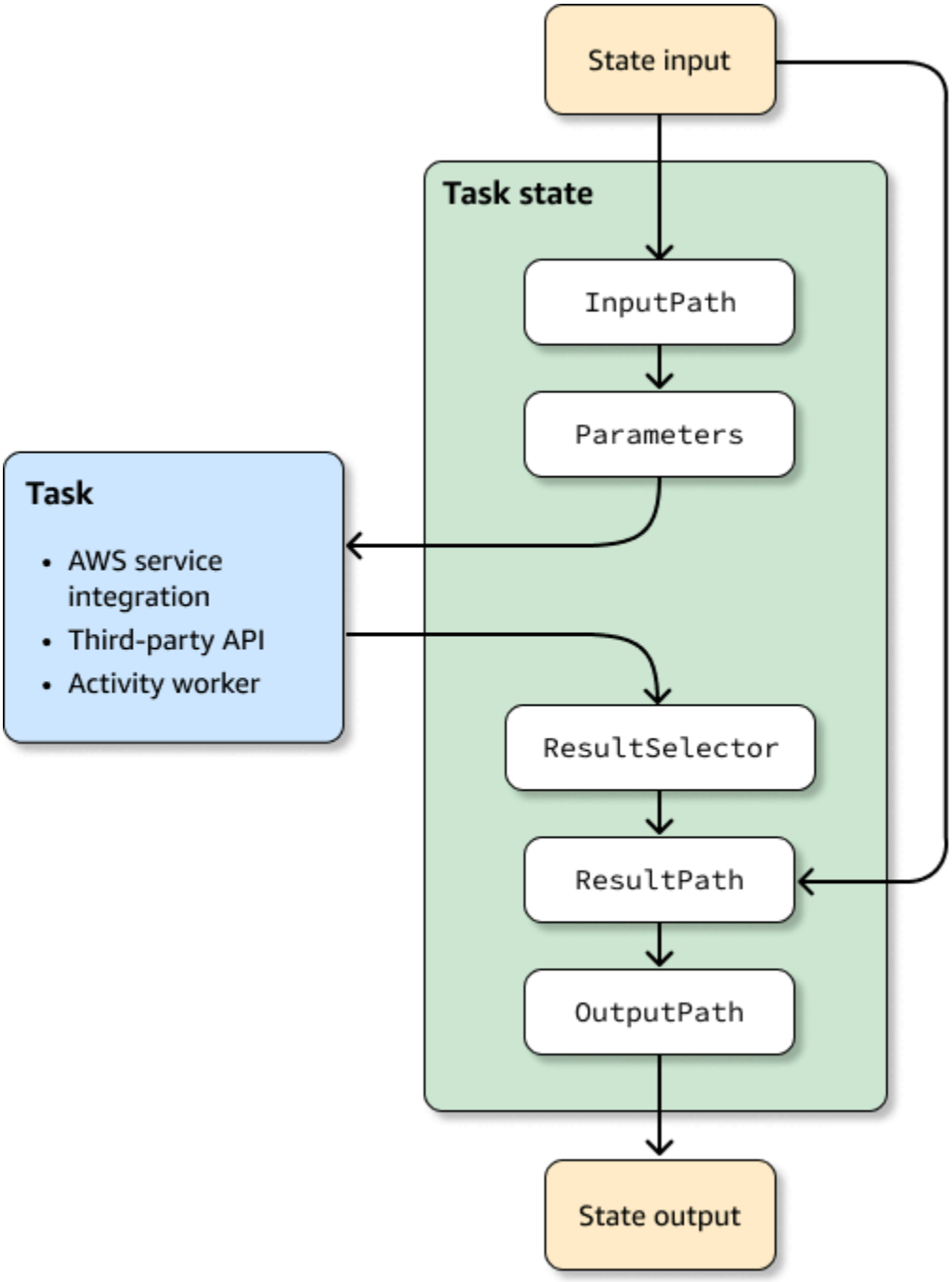
Menguji dan men-debug aliran data input dan output

`TestState`API berguna untuk menguji dan men-debug data yang mengalir melalui alur kerja Anda. Bagian ini memberikan beberapa konsep kunci dan menjelaskan cara menggunakan TestState untuk tujuan ini.

Konsep utama

DalamStep Functions, proses penyaringan dan manipulasi data JSON saat melewati status di mesin negara Anda disebut pemrosesan input dan output. Untuk informasi tentang cara kerjanya, lihatPengolahan Input dan Output di Step Functions.

Semua jenis status dalam Amazon States Language (ASL) (Tugas, Paralel, Peta, Lulus, Tunggu, Pilihan, Sukses, dan Gagal) berbagi satu set bidang umum untuk memfilter dan memanipulasi data JSON yang melewatinya. Bidang-bidang ini adalah: InputPathParameter,ResultSelector,,ResultPath, danOutputPath. Support untuk setiap bidang bervariasi di seluruh negara bagian. Saat runtime, Step Functions terapkan setiap bidang dalam urutan tertentu. Diagram berikut menunjukkan urutan bidang ini diterapkan ke data di dalam status Tugas:



Daftar berikut menjelaskan urutan penerapan bidang pemrosesan input dan output yang ditunjukkan pada diagram.

1. Input status adalah data JSON yang diteruskan ke keadaan saat ini dari keadaan sebelumnya.
2. InputPathmenyaring sebagian dari input status mentah.
3. Parametermengkonfigurasi set nilai untuk diteruskan ke Tugas.
4. Tugas melakukan pekerjaan dan mengembalikan hasilnya.

5. ResultSelectormemilih satu set nilai untuk menjaga dari hasil tugas.
6. ResultPathmenggabungkan hasil dengan input status mentah, atau menggantikan hasilnya dengan itu.
7. OutputPathmenyaring sebagian output untuk diteruskan ke keadaan berikutnya.
8. Output status adalah data JSON yang diteruskan dari keadaan saat ini ke keadaan berikutnya.

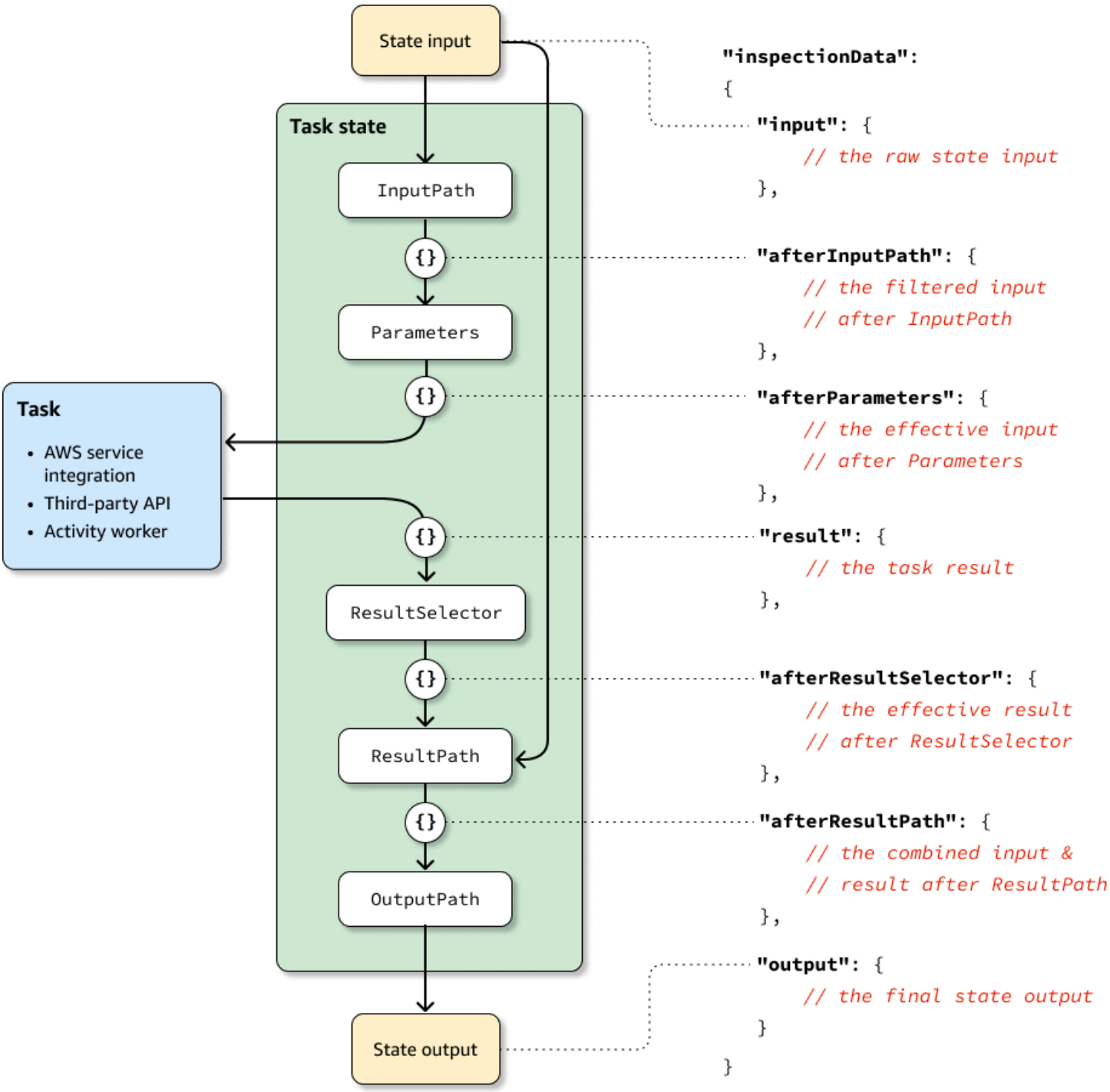
Bidang pemrosesan input dan output ini bersifat opsional. Jika Anda tidak menggunakan salah satu bidang ini dalam definisi status Anda, tugas akan menggunakan input status mentah, dan mengembalikan hasil tugas sebagai output status.

Menggunakan TestState untuk memeriksa pemrosesan input dan output

Saat Anda memanggil `TestState` API dan menyetel `inspectionLevel` parameternya`DEBUG`, respons API menyertakan objek yang dipanggil`inspectionData`. Objek ini berisi bidang untuk membantu Anda memeriksa bagaimana data disaring atau dimanipulasi dalam status saat dijalankan. Contoh berikut menunjukkan `inspectionData` objek untuk status Tugas.

```
"inspectionData":   {
  "input": string,
  "afterInputPath": string,
  "afterParameters": string,
  "result": string,
  "afterResultSelector": string,
  "afterResultPath": string,
  "output": string
}
```

Dalam contoh ini, setiap bidang yang berisi `after` awalan, menunjukkan data setelah bidang tertentu diterapkan. Misalnya, `afterInputPath` menunjukkan efek penerapan `InputPath` bidang untuk memfilter input status mentah. Diagram berikut memetakan setiap bidang definisi ASL ke bidang yang sesuai dalam `inspectionData` objek:



Untuk contoh penggunaan TestState API untuk men-debug pemrosesan input dan output, lihat berikut ini:

- Menguji status menggunakan tingkat inspeksi DEBUG di konsol Step Functions
- Menguji status menggunakan tingkat inspeksi DEBUG di AWS CLI

Menguji mesin negara secara lokal

AWSStep Functions Local adalah versi Step Functions yang dapat diunduh yang memungkinkan Anda mengembangkan dan menguji aplikasi menggunakan versi Step Functions yang berjalan di lingkungan pengembangan Anda sendiri. Versi lokal Step Functions dapat memanggil AWS Lambda fungsi, baik dalam AWS maupun saat berjalan secara lokal. Anda juga dapat mengoordinasikan layanan AWS yang didukung.

> **Note**
>
> Step Functions Local menggunakan akun dummy untuk bekerja.

Saat menjalankan Step Functions Local, Anda dapat menggunakan salah satu cara berikut untuk memanggil integrasi layanan:

- Mengkonfigurasi titik akhir lokal untuk AWS Lambda dan layanan lainnya. Untuk informasi tentang titik akhir yang didukung, lihatMengatur Opsi Konfigurasi untuk Step Functions Lokal.
- Melakukan panggilan langsung ke AWS layanan dari Step Functions Local.
- Mengejek respons dari integrasi layanan. Untuk informasi tentang menggunakan integrasi layanan tiruan, lihat. Menggunakan Integrasi Layanan Mocked

AWSStep Functions Local tersedia sebagai paket JAR atau image Docker mandiri yang berjalan di Microsoft Windows, Linux, macOS, dan platform lain yang mendukung Java atau Docker.

> **Warning**
>
> Versi download dimaksudkan untuk digunakan hanya untuk pengujian dan tidak boleh digunakan untuk memproses informasi sensitif. AWS Step Functions

> **Tip**
>
> Pastikan Anda menggunakan Step Functions Local versi 1.12.0 atau lebih tinggi untuk dapat menyertakan semua fungsi intrinsik dalam alur kerja Anda.

Topik berikut menjelaskan bagaimana Anda dapat mengatur Step Functions Local menggunakan file Docker dan JAR, dan menjalankan Step Functions Local untuk bekerja denganAWS Lambda, AWS Serverless Application Model (AWS SAM) CLI Local, atau layanan lain yang didukung.

Topik

- Menyiapkan Step Functions Lokal (Versi yang Dapat Diunduh) dan Docker
- Menyiapkan Step Functions Lokal (Versi yang Dapat Diunduh) - Versi Java
- Mengatur Opsi Konfigurasi untuk Step Functions Lokal
- Menjalankan Step Functions Lokal di Komputer Anda
- Pengujian Fungsi Langkah dan AWS SAM CLI lokal
- Menggunakan Integrasi Layanan Mocked

Menyiapkan Step Functions Lokal (Versi yang Dapat Diunduh) dan Docker

Citra Docker Step Functions Local memungkinkan Anda memulai dengan Step Functions Local cepat dengan menggunakan citra Docker terhadap semua dependensi yang dibutuhkan. Image Docker memungkinkan Anda untuk menyertakan Step Functions Local dalam build container Anda dan sebagai bagian dari pengujian integrasi berkelanjutan Anda.

Untuk mendapatkan image Docker untuk Step Functions Local, lihat https://hub.docker.com/r/amazon/ aws-stepfunctions-local, atau masukkan perintah Docker `pull` berikut.

```
docker pull amazon/aws-stepfunctions-local
```

Untuk memulai versi Step Functions yang dapat diunduh di Docker, jalankan perintah Docker berikut `run`

```
docker run -p 8083:8083 amazon/aws-stepfunctions-local
```

Untuk berinteraksi dengan AWS Lambda atau layanan lain yang didukung, Anda perlu mengonfigurasi kredensyal dan opsi konfigurasi lainnya terlebih dahulu. Untuk informasi lain, lihat topik berikut:

- Mengatur Opsi Konfigurasi untuk Step Functions Lokal
- Kredensial dan konfigurasi untuk Docker

Menyiapkan Step Functions Lokal (Versi yang Dapat Diunduh) - Versi Java

Versi yang dapat diunduh AWS Step Functions disediakan sebagai file JAR yang dapat dieksekusi dan sebagai gambar Docker. Aplikasi Java berjalan pada Windows, Linux, macOS, dan platform lain yang mendukung Java. Selain Java, Anda harus menginstal AWS Command Line Interface(AWS CLI). Untuk informasi tentang menginstal dan mengonfigurasi AWS CLI, lihat Panduan Pengguna AWS Command Line Interface.

Untuk menyiapkan dan menjalankan Step Functions di komputer Anda

1. Unduh Step Functions menggunakan tautan berikut.

Unduh Tautan	Checksum
.tar.gz	.tar.gz.md5
.zip	.zip.md5

2. Ekstrak file `.zip`.
3. Menguji unduhan dan melihat informasi versi.

```
$ java -jar StepFunctionsLocal.jar -v
Step Function Local
Version: 1.0.0
Build: 2019-01-21
```

4. (Opsional) Melihat daftar perintah yang tersedia.

```
$ java -jar StepFunctionsLocal.jar -h
```

5. Untuk memulai Step Functions di komputer Anda, buka command prompt, navigasikan ke direktori tempat Anda mengekstrak`StepFunctionsLocal.jar`, dan masukkan perintah berikut.

```
java -jar StepFunctionsLocal.jar
```

6. Untuk mengakses Step Functions yang berjalan secara lokal, gunakan parameter `--endpoint-url`. Misalnya, menggunakanAWS CLI, Anda akan menentukan perintah Step Functions sebagai berikut:

```
aws stepfunctions --endpoint-url http://localhost:8083 command
```

> **Note**
>
> Secara default, Step Functions Local menggunakan akun pengujian lokal dan kredensialnya, dan AWS Region diatur ke US East (Virginia N.). Untuk menggunakan Step Functions Local dengan AWS Lambda, atau layanan yang didukung lainnya, Anda harus mengonfigurasi kredensial dan Wilayah Anda.
> Jika Anda menggunakan alur kerja Express dengan Step Functions Local, riwayat eksekusi akan disimpan dalam file log. Itu tidak masuk ke CloudWatch Log. Jalur file log akan didasarkan pada grup CloudWatch log Log ARN yang disediakan saat Anda membuat mesin status lokal. Berkas log akan disimpan di `/aws/states/log-group-name/${execution_arn}.log` relatif terhadap lokasi tempat Anda menjalankan Step Functions Local. Misalnya, jika ARN eksekusi adalah:
>
> ```
> arn:aws:states:us-east-1:123456789012:express:test:example-ExpressLogGroup-wJalrXUtnFEMI
> ```
>
> berkas log adalah:
>
> ```
> aws/states/log-group-name/arn:aws:states:us-east-1:123456789012:express:test:example-ExpressLogGroup-wJalrXUtnFEMI.log
> ```

Mengatur Opsi Konfigurasi untuk Step Functions Lokal

Saat Anda memulai AWS Step Functions Local dengan menggunakan file JAR, Anda dapat mengatur opsi konfigurasi dengan menggunakan AWS Command Line Interface (AWS CLI), atau dengan memasukkannya ke dalam lingkungan sistem. Untuk Docker, Anda harus menentukan pilihan ini dalam file yang Anda referensi ketika memulai Step Functions Local.

Opsi konfigurasi

Saat Anda mengonfigurasi container Step Functions Local untuk menggunakan endpoint override seperti Lambda Endpoint dan Batch Endpoint, dan melakukan panggilan ke endpoint tersebut, Step

Functions Local tidak menggunakan kredensyal yang Anda tentukan. Menyetel penggantian titik akhir ini adalah opsional.

Opsi	Baris perintah	Environment
Akun	-akun, --aws-akun	AWS_ACCOUNT_ID
wilayah	-wilayah, --aws-region	AWS_DEFAULT_REGION
Skala Waktu Tunggu	-waitTimeScale, --wait-time-scale	WAIT_TIME_SCALE
Titik Akhir Lambda	-LambdaendPoint, --lambda-titik akhir	LAMBDA_ENDPOINT
Titik Akhir Batch	-BatchEndPoint, --batch-titik akhir	BATCH_ENDPOINT
Titik Akhir DynamoDB	-DynamodBendPoint, --dynamodb-titik akhir	DYNAMODB_ENDPOINT
Titik akhir ECS	-ECSendPoint, --ecs-titik akhir	ECS_ENDPOINT
Titik Akhir Glue	-GlueEndpoint, --lem-titik akhir	GLUE_ENDPOINT
SageMaker Titik akhir	-sageMakerEndpoint, --sagemaker-titik akhir	SAGE_MAKER_ENDPOINT
Titik akhir SQS	-SqSendPoint, --sqs-titik akhir	SQS_ENDPOINT
Titik Akhir SNS	-SnsendPoint, --sns-titik akhir	SNS_ENDPOINT
Titik akhir Step Functions	-stepFunctionsEndpoint, --step-functions-endpoint	STEP_FUNCTIONS_END POINT

Kredensial dan konfigurasi untuk Docker

Untuk mengonfigurasi Step Functions Local untuk Docker, buat file berikut: `aws-stepfunctions-local-credentials.txt`.

File ini berisi kredensyal Anda dan opsi konfigurasi lainnya. Berikut ini dapat digunakan sebagai template saat membuat `aws-stepfunctions-local-credentials.txt` file.

```
AWS_DEFAULT_REGION=AWS_REGION_OF_YOUR_AWS_RESOURCES
AWS_ACCESS_KEY_ID=YOUR_AWS_ACCESS_KEY
AWS_SECRET_ACCESS_KEY=YOUR_AWS_SECRET_KEY
WAIT_TIME_SCALE=VALUE
LAMBDA_ENDPOINT=VALUE
BATCH_ENDPOINT=VALUE
DYNAMODB_ENDPOINT=VALUE
ECS_ENDPOINT=VALUE
GLUE_ENDPOINT=VALUE
SAGE_MAKER_ENDPOINT=VALUE
SQS_ENDPOINT=VALUE
SNS_ENDPOINT=VALUE
STEP_FUNCTIONS_ENDPOINT=VALUE
```

Setelah Anda mengonfigurasi kredensial dan opsi konfigurasi Anda di `aws-stepfunctions-local-credentials.txt`, mulai Step Functions dengan perintah berikut.

```
docker run -p 8083:8083 --env-file aws-stepfunctions-local-credentials.txt amazon/aws-stepfunctions-local
```

> **Note**
>
> Disarankan untuk menggunakan nama DNS khusushost.docker.internal, yang menyelesaikan ke alamat IP internal yang digunakan host, seperti. `http://host.docker.internal:8000` Untuk informasi selengkapnya, lihat dokumentasi Docker untuk Mac dan Windows di fitur Jaringan di Docker Desktop untuk Mac dan fitur Jaringan di Docker Desktop untuk Windows masing-masing.

Menjalankan Step Functions Lokal di Komputer Anda

Gunakan Step Functions versi lokal untuk mengonfigurasi, mengembangkan, dan menguji mesin status di komputer Anda.

Jalankan mesin HelloWorld status secara lokal

Setelah Anda menjalankan Step Functions Local denganAWS Command Line Interface (AWS CLI), Anda dapat memulai eksekusi mesin status.

1. Buat mesin status dari AWS CLI dengan melarikan diri dari definisi mesin status.

```
aws stepfunctions --endpoint-url http://localhost:8083 create-state-machine --
definition "{\
  \"Comment\": \"A Hello World example of the Amazon States Language using a Pass
 state\",\
  \"StartAt\": \"HelloWorld\",\
  \"States\": {\
    \"HelloWorld\": {\
      \"Type\": \"Pass\",\
      \"End\": true\
    }\
  }}" --name "HelloWorld" --role-arn "arn:aws:iam::012345678901:role/DummyRole"
```

> **Note**
>
> `role-arn`Ini tidak digunakan untuk Step Functions Local, tetapi Anda harus memasukkannya dengan sintaks yang tepat. Anda dapat menggunakan Amazon Resource Name (ARN) dari contoh sebelumnya.

Jika Anda berhasil membuat mesin status, Step Functions merespons dengan tanggal penciptaan dan ARN mesin status.

```
{
    "creationDate": 1548454198.202,
    "stateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:HelloWorld"
}
```

2. Mulai eksekusi menggunakan ARN dari mesin status yang Anda buat.

```
aws stepfunctions --endpoint-url http://localhost:8083 start-execution --state-
machine-arn arn:aws:states:us-east-1:123456789012:stateMachine:HelloWorld
```

Step Functions Local dengan CLI Local AWS SAM

Anda dapat menggunakan versi lokal Step Functions dengan versi lokal AWS Lambda. Untuk mengonfigurasi ini, Anda harus memasang dan mengonfigurasi AWS SAM.

Untuk informasi selengkapnya tentang mengonfigurasi dan menjalankan AWS SAM, lihat berikut:

- Mengatur AWS SAM
- Mulai AWS SAM CLI Lokal

Ketika Lambda berjalan pada sistem lokal Anda, Anda dapat memulai Step Functions Local. Dari direktori tempat Anda mengekstrak file JAR lokal Step Functions, mulai Step Functions Local dan gunakan `--lambda-endpoint` parameter untuk mengonfigurasi titik akhir Lambda lokal.

```
java -jar StepFunctionsLocal.jar --lambda-endpoint http://127.0.0.1:3001 command
```

Untuk informasi selengkapnya tentang menjalankan Step Functions Local dengan AWS Lambda, lihat Pengujian Fungsi Langkah dan AWS SAM CLI lokal.

Pengujian Fungsi Langkah dan AWS SAM CLI lokal

Dengan AWS Step Functions dan AWS Lambda berjalan di mesin lokal Anda, Anda dapat menguji mesin status dan fungsi Lambda tanpa men-deploy kode Anda ke AWS.

Untuk informasi lain, lihat topik berikut:

- Menguji mesin negara secara lokal
- Mengatur AWS SAM

Topik

- Langkah 1: Siapkan AWS SAM
- Langkah 2: Uji coba AWS SAM CLI Local
- Langkah 3: Mulai AWS SAM CLI Local
- Langkah 4: Mulai Step Functions Local
- Langkah 5: Buat Mesin Status yang Mereferensikan Fungsi AWS SAM CLI Local Anda
- Langkah 6: Mulai Eksekusi Mesin Status Lokal Anda.

Langkah 1: Siapkan AWS SAM

CLI Lokal AWS Serverless Application Model (AWS SAM) memerlukan AWS Command Line Interface, AWS SAM, dan Docker agar dapat diinstal.

1. Instal CLI AWS SAM.

   > **Note**
   >
   > Sebelum menginstal AWS SAM CLI, Anda harus menginstal AWS CLI dan Docker. Lihat Prasyarat menginstal CLI AWS SAM.

2. Buka dokumentasi Quick Start AWS SAM. Pastikan untuk mengikuti langkah-langkah berikut:

   1. Inisialisasi Aplikasi
   2. Uji Aplikasi Secara Lokal

   Hal ini menciptakan direktori `sam-app`, dan membangun lingkungan yang mencakup fungsi Hello World Lambda berbasis Python.

Langkah 2: Uji coba AWS SAM CLI Local

Sekarang setelah Anda menginstal AWS SAM dan membuat fungsi Hello World Lambda, Anda dapat menguji fungsinya. Di `sam-app` direktori, masukkan perintah berikut:

```
sam local start-api
```

Langkah ini meluncurkan instans lokal dari fungsi Lambda Anda. Anda akan melihat output simillar sebagai berikut:

```
2019-01-31 16:40:27 Found credentials in shared credentials file: ~/.aws/credentials
2019-01-31 16:40:27 Mounting HelloWorldFunction at http://127.0.0.1:3000/hello [GET]
2019-01-31 16:40:27 You can now browse to the above endpoints to invoke your functions.
 You do not need to restart/reload SAM CLI while working on your functions changes will
 be reflected instantly/automatically. You only need to restart SAM CLI if you update
 your AWS SAM template
2019-01-31 16:40:27  * Running on http://127.0.0.1:3000/ (Press CTRL+C to quit)
```

Buka browser dan masukkan yang berikut ini:

```
http://127.0.0.1:3000/hello
```

Ini akan menampilkan simillar respon sebagai berikut:

```
{"message": "hello world", "location": "72.21.198.66"}
```

Masukkan CTRL+C untuk mengakhiri API Lambda.

Langkah 3: Mulai AWS SAM CLI Local

Setelah Anda menguji kemampuan bekerja fungsi tersebut, mulai AWS SAM CLI Local. Di `sam-app` direktori, masukkan perintah berikut:

```
sam local start-lambda
```

Ini dimulai AWS SAM CLI lokal dan menyediakan endpoint untuk digunakan, mirip dengan output berikut:

```
2019-01-29 15:33:32 Found credentials in shared credentials file: ~/.aws/credentials
2019-01-29 15:33:32 Starting the Local Lambda Service. You can now invoke your Lambda
 Functions defined in your template through the endpoint.
2019-01-29 15:33:32  * Running on http://127.0.0.1:3001/ (Press CTRL+C to quit)
```

Langkah 4: Mulai Step Functions Local

File JAR

Jika Anda menggunakan versi `.jar` file Step Functions Local, mulai Step Functions dan tentukan endpoint Lambda. Di direktori tempat Anda mengekstrak `.jar` file, masukkan perintah berikut:

```
java -jar StepFunctionsLocal.jar --lambda-endpoint http://localhost:3001
```

Saat Step Functions Local dimulai, ia memeriksa lingkungan, lalu kredensial dikonfigurasikan di file `~/.aws/credentials` Anda. Secara default, itu mulai menggunakan ID pengguna fiktif, dan terdaftar sebagai. `region us-east-1`

```
2019-01-29 15:38:06.324: Failed to load credentials from environment because Unable to
 load AWS credentials from environment variables (AWS_ACCESS_KEY_ID (or AWS_ACCESS_KEY)
 and AWS_SECRET_KEY (or AWS_SECRET_ACCESS_KEY))
2019-01-29 15:38:06.326: Loaded credentials from profile: default
2019-01-29 15:38:06.326: Starting server on port 8083 with account 123456789012, region
 us-east-1
```

Docker

Jika Anda menggunakan versi Docker dari Step Functions Local, luncurkan Step Functions dengan perintah berikut:

```
docker run -p 8083:8083 amazon/aws-stepfunctions-local
```

Untuk informasi tentang menginstal Step Functions versi Docker, lihat Menyiapkan Step Functions Lokal (Versi yang Dapat Diunduh) dan Docker.

> **Note**
>
> Anda dapat menentukan titik akhir melalui baris perintah atau dengan menetapkan variabel lingkungan jika Anda meluncurkan Step Functions dari file `.jar`. Untuk versi Docker, Anda harus menentukan titik akhir dan kredensial dalam file teks. Lihat Mengatur Opsi Konfigurasi untuk Step Functions Lokal.

Langkah 5: Buat Mesin Status yang Mereferensikan Fungsi AWS SAM CLI Local Anda

Setelah Step Functions Local berjalan, buat mesin state yang mereferensikan `HelloWorldFunction` yang Anda inisialisasi. Langkah 1: Siapkan AWS SAM

```
aws stepfunctions --endpoint http://localhost:8083 create-state-machine --definition
 "{\
  \"Comment\": \"A Hello World example of the Amazon States Language using an AWS
 Lambda Local function\",\
  \"StartAt\": \"HelloWorld\",\
  \"States\": {\
    \"HelloWorld\": {\
      \"Type\": \"Task\",\
      \"Resource\": \"arn:aws:lambda:us-east-1:123456789012:function:HelloWorldFunction
\",\
```

```
      \"End\": true\
    }\
  }\
}\
}}" --name "HelloWorld" --role-arn "arn:aws:iam::012345678901:role/DummyRole"
```

Tindakan ini akan menciptakan sebuah mesin status dan menyediakan Amazon Resource Name (ARN) yang dapat Anda gunakan untuk memulai eksekusi.

```
{
    "creationDate": 1548805711.403,
    "stateMachineArn": "arn:aws:states:us-east-1:123456789012:stateMachine:HelloWorld"
}
```

Langkah 6: Mulai Eksekusi Mesin Status Lokal Anda.

Setelah Anda membuat mesin negara, mulai eksekusi. Anda harus mereferensikan ARN endpoint dan state machine saat menggunakan perintah berikut: **`aws stepfunctions`**

```
aws stepfunctions --endpoint http://localhost:8083 start-execution --state-machine
 arn:aws:states:us-east-1:123456789012:stateMachine:HelloWorld --name test
```

Ini memulai eksekusi bernama `test` mesin `HelloWorld` negara Anda.

```
{
    "startDate": 1548810641.52,
    "executionArn": "arn:aws:states:us-east-1:123456789012:execution:HelloWorld:test"
}
```

Kini Step Functions berjalan secara lokal, Anda dapat berinteraksi dengan menggunakan AWS CLI. Misalnya, untuk mendapatkan informasi tentang eksekusi ini, gunakan perintah berikut:

```
aws stepfunctions --endpoint http://localhost:8083 describe-execution --execution-arn
 arn:aws:states:us-east-1:123456789012:execution:HelloWorld:test
```

`describe-execution`Memanggil eksekusi memberikan rincian yang lebih lengkap, mirip dengan output berikut:

```
{
    "status": "SUCCEEDED",
    "startDate": 1549056334.073,
```

```
    "name": "test",
    "executionArn": "arn:aws:states:us-east-1:123456789012:execution:HelloWorld:test",
    "stateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:HelloWorld",
    "stopDate": 1549056351.276,
    "output": "{\"statusCode\": 200, \"body\": \"{\\\"message\\\": \\\"hello world\\\",
 \\\"location\\\": \\\"72.21.198.64\\\"}\"}",
    "input": "{}"
}
```

Menggunakan Integrasi Layanan Mocked

Di Step Functions Local, Anda dapat menguji jalur eksekusi mesin status Anda tanpa benar-benar memanggil layanan terintegrasi dengan menggunakan integrasi layanan tiruan. Untuk mengonfigurasi mesin status Anda agar menggunakan integrasi layanan tiruan, Anda membuat file konfigurasi tiruan. Dalam file ini, Anda mendefinisikan output yang diinginkan dari integrasi layanan Anda sebagai respons yang diejek dan eksekusi yang menggunakan respons tiruan Anda untuk mensimulasikan jalur eksekusi sebagai kasus uji.

Dengan menyediakan file konfigurasi tiruan ke Step Functions Local, Anda dapat menguji panggilan integrasi layanan dengan menjalankan state machine yang menggunakan respons tiruan yang ditentukan dalam kasus pengujian alih-alih membuat panggilan integrasi layanan yang sebenarnya.

> **Note**
>
> Jika Anda tidak menentukan respons integrasi layanan tiruan dalam file konfigurasi tiruan, Step Functions Local akan memanggil integrasi AWS layanan menggunakan titik akhir yang Anda konfigurasikan saat menyiapkan Step Functions Local. Untuk informasi tentang mengonfigurasi titik akhir untuk Step Functions Local, lihat. Mengatur Opsi Konfigurasi untuk Step Functions Lokal

Topik

- Konsep kunci dalam topik ini
- Langkah 1: Tentukan Integrasi Layanan Mocked dalam File Konfigurasi Mock
- Langkah 2: Berikan File Konfigurasi Mock ke Step Functions Local
- Langkah 3: Jalankan Tes Integrasi Layanan Mocked
- File Konfigurasi untuk Integrasi Layanan Mocked

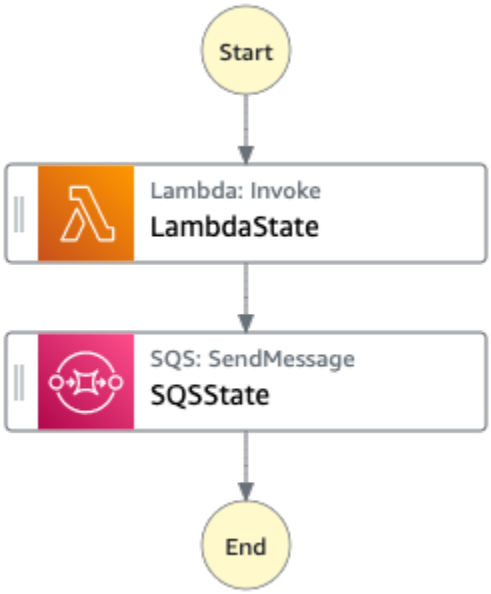
Konsep kunci dalam topik ini

Topik ini menggunakan beberapa konsep yang didefinisikan dalam daftar berikut:

- Integrasi Layanan Mocked - Mengacu pada status Tugas yang dikonfigurasi untuk menggunakan respons yang diejek alih-alih melakukan panggilan layanan yang sebenarnya.
- Respons Mocked - Mengacu pada data tiruan yang status Tugas dapat dikonfigurasi untuk digunakan.
- Kasus Uji - Mengacu pada eksekusi mesin status yang dikonfigurasi untuk menggunakan integrasi layanan yang diejek.
- File Konfigurasi Mock - Mengacu pada file konfigurasi tiruan yang berisi JSON, yang mendefinisikan integrasi layanan tiruan, respons yang diejek, dan kasus uji.

Langkah 1: Tentukan Integrasi Layanan Mocked dalam File Konfigurasi Mock

Anda dapat menguji Step Functions AWS SDK dan integrasi layanan yang dioptimalkan menggunakan Step Functions Local. Gambar berikut menunjukkan mesin status yang ditentukan dalam tab Definisi mesin Negara:



Untuk melakukan ini, Anda harus membuat file konfigurasi tiruan yang berisi bagian seperti yang didefinisikan dalamMemperkenalkan struktur konfigurasi tiruan.

1. Buat file bernama `MockConfigFile.json` untuk mengonfigurasi pengujian dengan integrasi layanan tiruan.

Contoh berikut menunjukkan file konfigurasi tiruan yang mereferensikan mesin negara dengan dua status yang ditentukan bernama `LambdaState` dan. `SQSState`

Mock configuration file example

Berikut ini adalah contoh file konfigurasi tiruan yang menunjukkan cara mengejek tanggapan dari menjalankan fungsi Lambda dan mengirim pesan ke Amazon SQS. Dalam contoh ini, mesin `LambdaSQSIntegration`negara berisi tiga kasus uji bernama`HappyPath`,`RetryPath`, dan `HybridPath` yang mengejek `Task` status bernama `LambdaState` dan`SQSState`. Negara-negara bagian ini menggunakan`MockedLambdaSuccess`,`MockedSQSSuccess`, dan tanggapan layanan yang `MockedLambdaRetry` diejek. Respons layanan yang diejek ini didefinisikan di `MockedResponses` bagian file.

```
{
  "StateMachines":{
    "LambdaSQSIntegration":{
      "TestCases":{
        "HappyPath":{
          "LambdaState":"MockedLambdaSuccess",
          "SQSState":"MockedSQSSuccess"
        },
        "RetryPath":{
          "LambdaState":"MockedLambdaRetry",
          "SQSState":"MockedSQSSuccess"
        },
        "HybridPath":{
          "LambdaState":"MockedLambdaSuccess"
        }
      }
    }
  },
  "MockedResponses":{
    "MockedLambdaSuccess":{
      "0":{
        "Return":{
          "StatusCode":200,
          "Payload":{
            "StatusCode":200,
            "body":"Hello from Lambda!"
          }
```

```
        }
      }
    },
    "LambdaMockedResourceNotReady":{
      "0":{
        "Throw":{
          "Error":"Lambda.ResourceNotReadyException",
          "Cause":"Lambda resource is not ready."
        }
      }
    },
    "MockedSQSSuccess":{
      "0":{
        "Return":{
          "MD5OfMessageBody":"3bcb6e8e-7h85-4375-b0bc-1a59812c6e51",
          "MessageId":"3bcb6e8e-8b51-4375-b0bc-1a59812c6e51"
        }
      }
    },
    "MockedLambdaRetry":{
      "0":{
        "Throw":{
          "Error":"Lambda.ResourceNotReadyException",
          "Cause":"Lambda resource is not ready."
        }
      },
      "1-2":{
        "Throw":{
          "Error":"Lambda.TimeoutException",
          "Cause":"Lambda timed out."
        }
      },
      "3":{
        "Return":{
          "StatusCode":200,
          "Payload":{
            "StatusCode":200,
            "body":"Hello from Lambda!"
          }
        }
      }
    }
  }
```

```
}
```

State machine definition

Berikut ini adalah contoh definisi mesin negara yang disebut`LambdaSQSIntegration`, yang mendefinisikan dua status tugas integrasi layanan bernama `LambdaState` dan`SQSState`. `LambdaState`berisi kebijakan coba lagi berdasarkan`States.ALL`.

```
{
  "Comment":"This state machine is called: LambdaSQSIntegration",
  "StartAt":"LambdaState",
  "States":{
    "LambdaState":{
      "Type":"Task",
      "Resource":"arn:aws:states:::lambda:invoke",
      "Parameters":{
        "Payload.$":"$",
        "FunctionName":"HelloWorldFunction"
      },
      "Retry":[
        {
          "ErrorEquals":[
            "States.ALL"
          ],
          "IntervalSeconds":2,
          "MaxAttempts":3,
          "BackoffRate":2
        }
      ],
      "Next":"SQSState"
    },
    "SQSState":{
      "Type":"Task",
      "Resource":"arn:aws:states:::sqs:sendMessage",
      "Parameters":{
        "QueueUrl":"https://sqs.us-east-1.amazonaws.com/123456789012/myQueue",
        "MessageBody.$":"$"
      },
      "End": true
    }
  }
}
```

Anda dapat menjalankan definisi mesin `LambdaSQSIntegration` status yang direferensikan dalam file konfigurasi tiruan menggunakan salah satu kasus uji berikut:

- `HappyPath`- Tes ini mengolok-olok output `LambdaState` dan `SQSState` menggunakan `MockedLambdaSuccess` dan `MockedSQSSuccess` masing-masing.
  - `LambdaState`Akan mengembalikan nilai berikut:

```
"0":{
  "Return":{
    "StatusCode":200,
    "Payload":{
      "StatusCode":200,
      "body":"Hello from Lambda!"
    }
  }
}
```

  - `SQSState`Akan mengembalikan nilai berikut:

```
"0":{
  "Return":{
    "MD5OfMessageBody":"3bcb6e8e-7h85-4375-b0bc-1a59812c6e51",
    "MessageId":"3bcb6e8e-8b51-4375-b0bc-1a59812c6e51"
  }
}
```

- `RetryPath`- Tes ini mengolok-olok output `LambdaState` dan `SQSState` menggunakan `MockedLambdaRetry` dan `MockedSQSSuccess` masing-masing. Selain itu, `LambdaState` dikonfigurasi untuk melakukan empat upaya coba lagi. Tanggapan yang diejek untuk upaya ini didefinisikan dan diindeks di negara bagian. `MockedLambdaRetry`
  - Upaya awal berakhir dengan kegagalan tugas yang berisi pesan penyebab dan kesalahan seperti yang ditunjukkan pada contoh berikut:

```
"0":{
  "Throw": {
    "Error": "Lambda.ResourceNotReadyException",
    "Cause": "Lambda resource is not ready."
  }
}
```

- Upaya percobaan ulang pertama dan kedua diakhiri dengan kegagalan tugas yang berisi pesan penyebab dan kesalahan seperti yang ditunjukkan pada contoh berikut:

```
"1-2":{
  "Throw": {
    "Error": "Lambda.TimeoutException",
    "Cause": "Lambda timed out."
  }
}
```

- Upaya coba lagi ketiga diakhiri dengan keberhasilan tugas yang berisi hasil status dari bagian Payload dalam respons Lambda yang diejek.

```
"3":{
  "Return": {
    "StatusCode": 200,
    "Payload": {
      "StatusCode": 200,
      "body": "Hello from Lambda!"
    }
  }
}
```

> **Note**
>
> - Untuk status dengan kebijakan coba lagi, Step Functions Local akan menghabiskan upaya percobaan ulang yang ditetapkan dalam kebijakan hingga menerima respons yang berhasil. Ini berarti bahwa Anda harus menunjukkan ejekan untuk percobaan ulang dengan nomor percobaan berturut-turut dan harus mencakup semua upaya coba lagi sebelum mengembalikan respons yang berhasil.
> - Jika Anda tidak menentukan respons yang diejek untuk upaya coba lagi tertentu, misalnya, coba lagi “3", eksekusi mesin status akan gagal.

- `HybridPath`- Tes ini mengolok-olok output dari. `LambdaState` Setelah berhasil `LambdaState` berjalan dan menerima data tiruan sebagai respons, `SQSState` melakukan panggilan layanan aktual ke sumber daya yang ditentukan dalam produksi.

Untuk informasi tentang cara memulai eksekusi pengujian dengan integrasi layanan tiruan, lihat. Langkah 3: Jalankan Tes Integrasi Layanan Mocked

2. Pastikan bahwa struktur respons yang diejek sesuai dengan struktur respons layanan aktual yang Anda terima saat Anda melakukan panggilan layanan terintegrasi. Untuk informasi tentang persyaratan struktural untuk tanggapan yang diejek, lihatMengkonfigurasi integrasi layanan yang diejek.

   Dalam contoh file konfigurasi tiruan sebelumnya, respons tiruan didefinisikan `MockedLambdaSuccess` dan `MockedLambdaRetry` sesuai dengan struktur respons aktual yang dikembalikan dari panggilan. `HelloFromLambda`

   > **Important**
   >
   > AWS tanggapan layanan dapat bervariasi dalam struktur antara layanan yang berbeda. Step Functions Local tidak memvalidasi jika struktur respons tiruan sesuai dengan struktur respons layanan yang sebenarnya. Anda harus memastikan bahwa tanggapan ejekan Anda sesuai dengan respons aktual sebelum pengujian. Untuk meninjau struktur respons layanan, Anda dapat melakukan panggilan layanan aktual menggunakan Step Functions atau melihat dokumentasi untuk layanan tersebut.

Langkah 2: Berikan File Konfigurasi Mock ke Step Functions Local

Anda dapat menyediakan file konfigurasi tiruan ke Step Functions Local dengan salah satu cara berikut:

Docker

> **Note**
>
> Jika Anda menggunakan versi Docker dari Step Functions Local, Anda dapat menyediakan file konfigurasi tiruan hanya menggunakan variabel lingkungan. Selain itu, Anda harus memasang file konfigurasi tiruan ke wadah Step Functions Local pada boot-up server awal.

Pasang file konfigurasi tiruan ke direktori apa pun dalam wadah Step Functions Local. Kemudian, atur variabel lingkungan bernama `SFN_MOCK_CONFIG` yang berisi path ke file konfigurasi tiruan dalam wadah. Metode ini memungkinkan file konfigurasi tiruan diberi nama apa saja selama variabel lingkungan berisi path file dan nama.

Perintah berikut menunjukkan format untuk memulai image Docker.

```
docker run -p 8083:8083
--mount type=bind,readonly,source={absolute path to mock config file},destination=/
home/StepFunctionsLocal/MockConfigFile.json
-e SFN_MOCK_CONFIG="/home/StepFunctionsLocal/MockConfigFile.json" amazon/aws-
stepfunctions-local
```

Contoh berikut menggunakan perintah untuk memulai image Docker.

```
docker run -p 8083:8083
--mount type=bind,readonly,source=/Users/admin/Desktop/workplace/
MockConfigFile.json,destination=/home/StepFunctionsLocal/MockConfigFile.json
-e SFN_MOCK_CONFIG="/home/StepFunctionsLocal/MockConfigFile.json" amazon/aws-
stepfunctions-local
```

JAR File

Gunakan salah satu cara berikut untuk menyediakan file konfigurasi tiruan ke Step Functions Local:

- Tempatkan file konfigurasi tiruan di direktori yang sama dengan. `Step FunctionsLocal.jar` Saat menggunakan metode ini, Anda harus memberi nama file `MockConfigFile.json` konfigurasi tiruan.
- Dalam sesi yang menjalankan Step Functions Local`SFN_MOCK_CONFIG`, atur variabel lingkungan bernama, ke path lengkap file konfigurasi tiruan. Metode ini memungkinkan file konfigurasi tiruan diberi nama apa saja selama variabel lingkungan berisi path file dan nama. Dalam contoh berikut, `SFN_MOCK_CONFIG` variabel diatur untuk menunjuk pada file konfigurasi tiruan bernama`EnvSpecifiedMockConfig.json`, terletak di `/home/workspace` direktori.

  ```
  export SFN_MOCK_CONFIG="/home/workspace/EnvSpecifiedMockConfig.json"
  ```

> **Note**
>
> - Jika Anda tidak menyediakan variabel lingkungan `SFN_MOCK_CONFIG` ke Step Functions Local, secara default, ia akan mencoba membaca file konfigurasi tiruan bernama `MockConfigFile.json` dalam direktori tempat Anda meluncurkan Step Functions Local.
> - Jika Anda menempatkan file konfigurasi tiruan di direktori yang sama dengan `Step FunctionsLocal.jar` dan mengatur variabel lingkunganSFN_MOCK_CONFIG, Step Functions Local akan membaca file yang ditentukan oleh variabel lingkungan.

Langkah 3: Jalankan Tes Integrasi Layanan Mocked

Setelah Anda membuat dan menyediakan file konfigurasi tiruan ke Step Functions Local, jalankan state machine yang dikonfigurasi dalam file konfigurasi tiruan menggunakan integrasi layanan tiruan. Kemudian periksa hasil eksekusi menggunakan tindakan API.

1. Buat mesin status berdasarkan definisi yang disebutkan sebelumnya dalam file konfigurasi tiruan.

```
aws stepfunctions create-state-machine \
    --endpoint http://localhost:8083 \
    --definition "{\"Comment\":\"Thisstatemachineiscalled:LambdaSQSIntegration
\",\"StartAt\":\"LambdaState\",\"States\":{\"LambdaState\":{\"Type\":
\"Task\",\"Resource\":\"arn:aws:states:::lambda:invoke\",\"Parameters
\":{\"Payload.$\":\"$\",\"FunctionName\":\"arn:aws:lambda:us-
east-1:123456789012:function:HelloWorldFunction\"},\"Retry\":[{\"ErrorEquals
\":[\"States.ALL\"],\"IntervalSeconds\":2,\"MaxAttempts\":3,\"BackoffRate
\":2}],\"Next\":\"SQSState\"},\"SQSState\":{\"Type\":\"Task\",\"Resource\":
\"arn:aws:states:::sqs:sendMessage\",\"Parameters\":{\"QueueUrl\":\"https://
sqs.us-east-1.amazonaws.com/123456789012/myQueue\",\"MessageBody.$\":\"$\"},\"End
\":true}}}" \
    --name "LambdaSQSIntegration" --role-arn "arn:aws:iam::123456789012:role/
service-role/LambdaSQSIntegration"
```

2. Jalankan mesin status menggunakan integrasi layanan tiruan.

   Untuk menggunakan file konfigurasi tiruan, lakukan panggilan `StartExecution` API pada mesin status yang dikonfigurasi dalam file konfigurasi tiruan. Untuk melakukan ini, tambahkan

sufiks,#*test_name*, ke mesin negara ARN yang digunakan oleh. `StartExecution` *test_name*adalah kasus uji, yang dikonfigurasi untuk mesin status dalam file konfigurasi tiruan yang sama.

Perintah berikut adalah contoh yang menggunakan `LambdaSQSIntegration` state machine dan konfigurasi tiruan. Dalam contoh ini, mesin `LambdaSQSIntegration` status dijalankan menggunakan `HappyPath` pengujian yang didefinisikan dalamLangkah 1: Tentukan Integrasi Layanan Mocked dalam File Konfigurasi Mock. `HappyPath`Pengujian berisi konfigurasi untuk eksekusi untuk menangani panggilan integrasi layanan tiruan yang dilakukan `LambdaState` dan `SQSState` status menggunakan respons layanan yang diejek `MockedLambdaSuccess` dan `MockedSQSSuccess` diejek.

```
aws stepfunctions start-execution \
    --endpoint http://localhost:8083 \
    --name executionWithHappyPathMockedServices \
    --state-machine arn:aws:states:us-
east-1:123456789012:stateMachine:LambdaSQSIntegration#HappyPath
```

3. Lihat respons eksekusi mesin status.

   Respons terhadap panggilan `StartExecution` menggunakan uji integrasi layanan tiruan sama dengan respons terhadap panggilan `StartExecution` secara normal, yang mengembalikan ARN eksekusi dan tanggal mulai.

   Berikut ini adalah contoh respons untuk memanggil `StartExecution` menggunakan uji integrasi layanan tiruan:

   ```
   {
      "startDate":"2022-01-28T15:03:16.981000-05:00",
      "executionArn":"arn:aws:states:us-
   east-1:123456789012:execution:LambdaSQSIntegration:executionWithHappyPathMockedServices"
   }
   ```

4. Periksa hasil eksekusi dengan melakukan panggilan`ListExecutions`,`DescribeExecution`, atau `GetExecutionHistory` API.

   ```
   aws stepfunctions get-execution-history \
       --endpoint http://localhost:8083 \
       --execution-arn arn:aws:states:us-
   east-1:123456789012:execution:LambdaSQSIntegration:executionWithHappyPathMockedServices
   ```

Contoh berikut menunjukkan bagian dari respon untuk memanggil `GetExecutionHistory` menggunakan ARN eksekusi dari contoh respon yang ditunjukkan pada langkah 2. Dalam contoh ini, output dari `LambdaState` dan `SQSState` adalah data tiruan yang didefinisikan dalam `MockedLambdaSuccess` dan `MockedSQSSuccess` dalam file konfigurasi tiruan. Selain itu, data tiruan digunakan dengan cara yang sama seperti data yang dikembalikan dengan melakukan panggilan integrasi layanan aktual akan digunakan. Juga, dalam contoh ini, output dari `LambdaState` diteruskan ke `SQSState` sebagai input.

```
{
    "events": [
        ...
        {
            "timestamp": "2021-12-02T19:39:48.988000+00:00",
            "type": "TaskStateEntered",
            "id": 2,
            "previousEventId": 0,
            "stateEnteredEventDetails": {
                "name": "LambdaState",
                "input": "{}",
                "inputDetails": {
                    "truncated": false
                }
            }
        },
        ...
        {
            "timestamp": "2021-11-25T23:39:10.587000+00:00",
            "type": "LambdaFunctionSucceeded",
            "id": 5,
            "previousEventId": 4,
            "lambdaFunctionSucceededEventDetails": {
                "output": "{\"statusCode\":200,\"body\":\"\\\"Hello from Lambda!\\
\"\"}",
                "outputDetails": {
                    "truncated": false
                }
            }
        },
        ...
            "timestamp": "2021-12-02T19:39:49.464000+00:00",
            "type": "TaskStateEntered",
```

```
            "id": 7,
            "previousEventId": 6,
            "stateEnteredEventDetails": {
                "name": "SQSState",
                "input": "{\"statusCode\":200,\"body\":\"\\\"Hello from Lambda!\\
\"\"}",
                "inputDetails": {
                    "truncated": false
                }
            }
        },
        ...
        {
            "timestamp": "2021-11-25T23:39:10.652000+00:00",
            "type": "TaskSucceeded",
            "id": 10,
            "previousEventId": 9,
            "taskSucceededEventDetails": {
                "resourceType": "sqs",
                "resource": "sendMessage",
                "output": "{\"MD5OfMessageBody\":\"3bcb6e8e-7h85-4375-
b0bc-1a59812c6e51\",\"MessageId\":\"3bcb6e8e-8b51-4375-b0bc-1a59812c6e51\"}",
                "outputDetails": {
                    "truncated": false
                }
            }
        },
        ...
    ]
}
```

File Konfigurasi untuk Integrasi Layanan Mocked

Untuk menggunakan integrasi layanan tiruan, Anda harus terlebih dahulu membuat file konfigurasi tiruan bernama `MockConfigFile.json` berisi konfigurasi tiruan Anda. Kemudian berikan Step Functions Local dengan file konfigurasi tiruan. File konfigurasi ini mendefinisikan kasus pengujian, yang berisi status tiruan yang menggunakan respons integrasi layanan yang diejek. Bagian berikut berisi informasi tentang struktur konfigurasi tiruan yang mencakup status tiruan dan tanggapan yang diejek:

Topik

- Memperkenalkan struktur konfigurasi tiruan
- Mengkonfigurasi integrasi layanan yang diejek

Memperkenalkan struktur konfigurasi tiruan

Konfigurasi tiruan adalah objek JSON yang berisi bidang tingkat atas berikut:

- `StateMachines`- Bidang objek ini mewakili mesin negara dikonfigurasi untuk menggunakan integrasi layanan mocked.
- `MockedResponse`- Bidang objek ini mewakili tanggapan mengejek untuk panggilan integrasi layanan.

Berikut ini adalah contoh file konfigurasi tiruan yang mencakup `StateMachine` definisi dan`MockedResponse`.

```
{
  "StateMachines":{
    "LambdaSQSIntegration":{
      "TestCases":{
        "HappyPath":{
          "LambdaState":"MockedLambdaSuccess",
          "SQSState":"MockedSQSSuccess"
        },
        "RetryPath":{
          "LambdaState":"MockedLambdaRetry",
          "SQSState":"MockedSQSSuccess"
        },
        "HybridPath":{
          "LambdaState":"MockedLambdaSuccess"
        }
      }
    }
  },
  "MockedResponses":{
    "MockedLambdaSuccess":{
      "0":{
        "Return":{
          "StatusCode":200,
          "Payload":{
            "StatusCode":200,
            "body":"Hello from Lambda!"
```

```
        }
      }
    }
  },
  "LambdaMockedResourceNotReady":{
    "0":{
      "Throw":{
        "Error":"Lambda.ResourceNotReadyException",
        "Cause":"Lambda resource is not ready."
      }
    }
  },
  "MockedSQSSuccess":{
    "0":{
      "Return":{
        "MD5OfMessageBody":"3bcb6e8e-7h85-4375-b0bc-1a59812c6e51",
        "MessageId":"3bcb6e8e-8b51-4375-b0bc-1a59812c6e51"
      }
    }
  },
  "MockedLambdaRetry":{
    "0":{
      "Throw":{
        "Error":"Lambda.ResourceNotReadyException",
        "Cause":"Lambda resource is not ready."
      }
    },
    "1-2":{
      "Throw":{
        "Error":"Lambda.TimeoutException",
        "Cause":"Lambda timed out."
      }
    },
    "3":{
      "Return":{
        "StatusCode":200,
        "Payload":{
          "StatusCode":200,
          "body":"Hello from Lambda!"
        }
      }
    }
  }
}
```

```
}
```

Referensi bidang konfigurasi tiruan

Bagian berikut menjelaskan bidang objek tingkat atas yang harus Anda tentukan dalam konfigurasi tiruan Anda.

- StateMachines
- MockedResponses

StateMachines

`StateMachines`Objek mendefinisikan mesin negara mana yang akan menggunakan integrasi layanan yang diejek. Konfigurasi untuk setiap mesin negara direpresentasikan sebagai bidang tingkat `StateMachines` atas. Nama field adalah nama mesin negara dan nilai adalah objek yang berisi satu bidang bernama`TestCases`, yang bidangnya mewakili kasus uji mesin negara.

Sintaks berikut menunjukkan mesin negara dengan dua kasus uji:

```
"MyStateMachine": {
  "TestCases": {
    "HappyPath": {
      ...
    },
    "SadPath": {
      ...
    }
  }
```

TestCases

Bidang `TestCases` mewakili kasus uji individu untuk mesin negara. Nama setiap kasus uji harus unik per mesin negara dan nilai setiap kasus uji adalah objek yang menentukan respons yang diejek untuk digunakan untuk status Tugas di mesin negara.

Contoh berikut dari `TestCase` link dua `Task` negara untuk dua`MockedResponses`:

```
"HappyPath": {
  "SomeTaskState": "SomeMockedResponse",
```

```
  "AnotherTaskState": "AnotherMockedResponse"
}
```

MockedResponses

`MockedResponses`adalah objek yang berisi beberapa objek respon yang diejek dengan nama bidang yang unik. Objek respons yang diejek mendefinisikan hasil yang berhasil atau keluaran kesalahan untuk setiap pemanggilan status Tugas yang diejek. Anda menentukan nomor pemanggilan menggunakan string integer individual, seperti “0", “1", “2", dan “3" atau rentang bilangan bulat inklusif, seperti “0-1", “2-3”.

Saat Anda mengejek Tugas, Anda harus menentukan respons yang diejek untuk setiap pemanggilan. Respons harus berisi satu bidang bernama `Return` atau `Throw` yang nilainya adalah hasil atau keluaran kesalahan untuk pemanggilan Tugas yang diejek. Jika Anda tidak menentukan respons yang diejek, eksekusi mesin negara akan gagal.

Berikut ini adalah contoh dari `MockedResponse` dengan `Throw` dan `Return` objek. Dalam contoh ini, tiga kali pertama mesin negara dijalankan, respon yang `"0-2"` ditentukan dalam dikembalikan, dan keempat kalinya mesin negara berjalan, respon yang `"3"` ditentukan dalam dikembalikan.

```
"SomeMockedResponse": {
  "0-2": {
    "Throw": {
      ...
    }
  },
  "3": {
    "Return": {
      ...
    }
  }
}
```

> **Note**
>
> Jika Anda menggunakan `Map` status, dan ingin memastikan tanggapan yang dapat diprediksi untuk `Map` status, tetapkan nilai `maxConcurrency` ke 1. Jika Anda menetapkan nilai yang lebih besar dari 1, Step Functions Local akan menjalankan beberapa iterasi secara bersamaan, yang akan menyebabkan urutan eksekusi keseluruhan status di seluruh iterasi menjadi tidak dapat diprediksi. Hal ini selanjutnya dapat menyebabkan Step Functions Local

menggunakan respons mocked yang berbeda untuk keadaan iterasi dari satu eksekusi ke eksekusi berikutnya.

Kembali

`Return`direpresentasikan sebagai bidang `MockedResponse` objek. Ini menentukan hasil yang sukses dari negara Tugas diejek.

Berikut ini adalah contoh `Return` objek yang berisi respons tiruan untuk memanggil `Invoke`fungsi Lambda:

```
"Return": {
  "StatusCode": 200,
  "Payload": {
    "StatusCode": 200,
    "body": "Hello from Lambda!"
  }
}
```

Lempar

`Throw`direpresentasikan sebagai bidang `MockedResponse` objek. Ini menentukan output kesalahan Tugas gagal. Nilai `Throw` harus berupa objek yang berisi `Error` dan `Cause` bidang dengan nilai string. Selain itu, nilai string yang Anda tentukan di `Error` bidang di `MockConfigFile.json` harus sesuai dengan kesalahan yang ditangani di `Retry` dan `Catch` bagian mesin negara Anda.

Berikut ini adalah contoh `Throw` objek yang berisi respons tiruan untuk memanggil `Invoke`fungsi Lambda:

```
"Throw": {
  "Error": "Lambda.TimeoutException",
  "Cause": "Lambda timed out."
}
```

Mengkonfigurasi integrasi layanan yang diejek

Anda dapat mengejek integrasi layanan apa pun menggunakan Step Functions Local. Namun, Step Functions Local tidak menerapkan mocks agar sama dengan API asli. Tugas yang diejek tidak akan pernah memanggil endpoint layanan. Jika Anda tidak menentukan respons yang diejek, Tugas

akan mencoba memanggil titik akhir layanan. Selain itu, Step Functions Local akan secara otomatis menghasilkan token tugas saat Anda mengejek Task menggunakan. `.waitForTaskToken`

Praktik terbaik untuk Step Functions

Praktik terbaik berikut untuk menerapkan alur kerja AWS Step Functions dapat membantu Anda mengoptimalkan performa implementasi Anda.

Topik

- Gunakan timeout untuk menghindari eksekusi macet
- Gunakan ARN Amazon S3 bukan meneruskan muatan besar
- Hindari mencapai kuota sejarah
- Menangani pengecualian layanan Lambda
- Hindari latensi saat polling untuk tugas aktivitas
- Memilih Alur Kerja Standar atau Ekspres
- Pembatasan ukuran kebijakan sumber daya Amazon CloudWatch Logs

Gunakan timeout untuk menghindari eksekusi macet

Secara default, Amazon States Language tidak menentukan batas waktu untuk definisi mesin status. Tanpa timeout eksplisit, Step Functions sering kali hanya bergantung pada respons dari seorang pekerja aktivitas untuk mengetahui bahwa tugas selesai. Jika terjadi kesalahan dan `TimeoutSeconds` bidang tidak ditentukan untuk `Task` status `Activity` atau, eksekusi macet menunggu respons yang tidak akan pernah datang.

Untuk menghindari situasi ini, tentukan batas waktu yang wajar saat Anda membuat mesin `Task` di negara bagian Anda. Misalnya:

```
"ActivityState": {
  "Type": "Task",
  "Resource": "arn:aws:states:us-east-1:123456789012:activity:HelloWorld",
  "TimeoutSeconds": 300,
  "Next": "NextState"
}
```

Jika Anda menggunakan callback dengan token tugas (. waitForTaskToken), kami menyarankan Anda menggunakan detak jantung dan menambahkan `HeartbeatSeconds` bidang dalam definisi `Task` status Anda. Anda dapat mengatur `HeartbeatSeconds` menjadi kurang dari batas waktu

tugas sehingga jika alur kerja Anda gagal dengan kesalahan detak jantung maka Anda tahu itu karena kegagalan tugas alih-alih tugas membutuhkan waktu lama untuk diselesaikan.

```
{
  "StartAt": "Push to SQS",
  "States": {
    "Push to SQS": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sqs:sendMessage.waitForTaskToken",
      "HeartbeatSeconds": 600,
      "Parameters": {
        "MessageBody": { "myTaskToken.$": "$$.Task.Token" },
        "QueueUrl": "https://sqs.us-east-1.amazonaws.com/123456789012/push-based-queue"
      },
      "ResultPath": "$.SQS",
      "End": true
    }
  }
}
```

Untuk informasi selengkapnya, lihat Tugas di dokumentasi Amazon States Language.

> **Note**
>
> Anda dapat mengatur timeout untuk mesin status Anda menggunakan kolom `TimeoutSeconds` di ketetapan Amazon States Language. Untuk informasi selengkapnya, lihat Struktur mesin status.

Gunakan ARN Amazon S3 bukan meneruskan muatan besar

Eksekusi yang meneruskan muatan data yang besar antar status dapat dihentikan. Jika data yang Anda lewati antar status mungkin bertambah hingga lebih dari 256 KB, gunakan Amazon Simple Storage Service (Amazon S3) untuk menyimpan data, dan mengurai Amazon Resource Name (ARN) bucket dalam `Payload` parameter untuk mendapatkan nama bucket dan nilai kunci. Atau, sesuaikan implementasi Anda sehingga Anda meneruskan muatan yang lebih kecil dalam eksekusi Anda.

Dalam contoh berikut, mesin status meneruskan input ke AWS Lambda fungsi, yang memproses file JSON di bucket Amazon S3. Setelah Anda menjalankan mesin status ini, fungsi Lambda membaca isi file JSON, dan mengembalikan konten file sebagai output.

Buat fungsi Lambda

Fungsi Lambda berikut bernama *pass-large-payload* membaca konten file JSON yang disimpan dalam bucket Amazon S3 tertentu.

> **Note**
>
> Setelah Anda membuat fungsi Lambda ini, pastikan Anda memberikan peran IAM-nya izin yang sesuai untuk membaca dari bucket Amazon S3. Misalnya, lampirkan ReadOnlyAccess izin AmazonS3 ke peran fungsi Lambda.

```
import json
import boto3
import io
import os

s3 = boto3.client('s3')

def lambda_handler(event, context):
    event = event['Input']
    final_json = str()

    s3 = boto3.resource('s3')
    bucket = event['bucket'].split(':')[-1]
    filename = event['key']
    directory = "/tmp/{}".format(filename)

    s3.Bucket(bucket).download_file(filename, directory)

    with open(directory, "r") as jsonfile:

        final_json = json.load(jsonfile)

    os.popen("rm -rf /tmp")

    return final_json
```

Buat mesin negara

Mesin status berikut memanggil fungsi Lambda yang sebelumnya Anda buat.

```
{
   "StartAt":"Invoke Lambda function",
   "States":{
      "Invoke Lambda function":{
         "Type":"Task",
         "Resource":"arn:aws:states:::lambda:invoke",
         "Parameters":{
            "FunctionName":"arn:aws:lambda:us-east-2:123456789012:function:pass-large-payload",
            "Payload":{
               "Input.$":"$"
            }
         },
         "OutputPath": "$.Payload",
         "End":true
      }
   }
}
```

Daripada meneruskan sejumlah besar data dalam input, Anda dapat menyimpan data tersebut di bucket Amazon S3, dan meneruskan Amazon Resource Name (ARN) bucket dalam `Payload` parameter untuk mendapatkan nama bucket dan nilai kunci. Fungsi Lambda Anda kemudian dapat menggunakan ARN tersebut untuk mengakses data secara langsung. Berikut ini adalah contoh input untuk eksekusi mesin status, di mana data disimpan `data.json` dalam bucket Amazon S3 bernama. *large-payload-json*

```
{
  "key": "data.json",
  "bucket": "arn:aws:s3:::large-payload-json"
}
```

Hindari mencapai kuota sejarah

AWS Step Functionsmemiliki kuota keras 25.000 entri dalam riwayat acara eksekusi. Ketika eksekusi mencapai 24.999 peristiwa, ia menunggu acara berikutnya terjadi.

- Jika nomor acara 25.000 adalah`ExecutionSucceeded`, eksekusi selesai dengan sukses.
- Jika nomor acara 25.000 tidak`ExecutionSucceeded`, `ExecutionFailed` peristiwa dicatat dan eksekusi mesin status gagal karena mencapai batas riwayat

Untuk menghindari mencapai kuota ini untuk eksekusi jangka panjang, Anda dapat mencoba salah satu solusi berikut:

- Gunakan status Peta dalam mode Terdistribusi. Dalam mode ini, `Map` status menjalankan setiap iterasi sebagai eksekusi alur kerja anak, yang memungkinkan konkurensi tinggi hingga 10.000 eksekusi alur kerja anak paralel. Setiap eksekusi alur kerja anak memiliki riwayat eksekusi terpisah sendiri dari alur kerja induk.
- Mulai eksekusi mesin status baru langsung dari `Task` keadaan eksekusi yang sedang berjalan. Untuk memulai eksekusi alur kerja bersarang seperti itu, gunakan tindakan `StartExecution` API Fungsi Langkah di mesin status induk bersama dengan parameter yang diperlukan. Untuk informasi selengkapnya tentang penggunaan alur kerja bersarang, lihat Mulai Eksekusi Alur Kerja dari Status Tugas. atau Menggunakan tindakan Step Functions API untuk melanjutkan tutorial eksekusi baru.

  > **Tip**
  >
  > Untuk menerapkan contoh alur kerja bersarang ke AndaAkun AWS, lihat Modul 13 - Alur Kerja Ekspres Bersarang.

- Terapkan pola yang menggunakan AWS Lambda fungsi yang dapat memulai eksekusi baru mesin status Anda untuk membagi pekerjaan yang sedang berlangsung di beberapa eksekusi alur kerja. Untuk informasi selengkapnya, lihat tutorial Menggunakan fungsi Lambda untuk melanjutkan eksekusi baru.

Menangani pengecualian layanan Lambda

AWS Lambda kadang-kadang dapat mengalami kesalahan layanan sementara. Dalam hal ini, memanggil Lambda menghasilkan kesalahan 500, `ClientExecutionTimeoutException` seperti,, `ServiceExceptionAWSLambdaException`, atau. `SdkClientException` Sebagai praktik terbaik, tangani pengecualian ini secara proaktif di mesin status Anda untuk `Retry` memanggil fungsi Lambda, atau untuk `Catch` kesalahan.

Kesalahan Lambda dilaporkan sebagai `Lambda.`*`ErrorName`*. Untuk mencoba lagi kesalahan pengecualian layanan Lambda, Anda dapat menggunakan kode `Retry` berikut.

```
"Retry": [ {
   "ErrorEquals": [ "Lambda.ClientExecutionTimeoutException",
 "Lambda.ServiceException", "Lambda.AWSLambdaException", "Lambda.SdkClientException"],
```

```
   "IntervalSeconds": 2,
   "MaxAttempts": 6,
   "BackoffRate": 2
} ]
```

> **Note**
>
> Kesalahan tidak tertangani di Lambda dilaporkan sebagai `Lambda.Unknown` pada output kesalahan. Ini termasuk out-of-memory kesalahan dan batas waktu fungsi. Anda dapat mencocokkan di `Lambda.Unknown`, `States.ALL`, atau `States.TaskFailed` untuk menangani kesalahan ini. Ketika Lambda mencapai jumlah maksimum permintaan, kesalahannya adalah `Lambda.TooManyRequestsException`. Untuk informasi selengkapnya tentang kesalahan fungsi Lambda, lihat Penanganan kesalahan dan percobaan ulang otomatis di Panduan PengembangAWS Lambda.

Untuk informasi selengkapnya, lihat yang berikut:

- Mencoba kembali setelah kesalahan
- Menangani kondisi kesalahan menggunakan mesin status Step Functions
- Kesalahan Pemanggilan Lambda

Hindari latensi saat polling untuk tugas aktivitas

API `GetActivityTask` dirancang untuk menyediakan `taskToken` satu kali. Jika `taskToken` dijatuhkan saat berkomunikasi dengan pekerja aktivitas, sejumlah permintaan `GetActivityTask` dapat diblokir selama 60 detik menunggu respons hingga `GetActivityTask` waktu habis.

Jika Anda hanya memiliki sejumlah kecil polling yang menunggu respons, mungkin semua permintaan akan mengantre di belakang permintaan yang diblokir dan berhenti. Namun, jika Anda memiliki sejumlah besar polling yang belum diproses untuk setiap aktivitas Amazon Resource Name (ARN), dan beberapa persentase permintaan Anda terjebak menunggu, akan ada banyak lagi yang masih bisa mendapatkan `taskToken` dan mulai memproses tugas.

Untuk sistem produksi, kami merekomendasikan setidaknya 100 polling terbuka per aktivitas ARN di setiap titik waktu. Jika satu polling diblokir, dan sebagian dari polling tersebut mengantre di belakangnya, masih banyak lagi permintaan yang akan menerima `taskToken` untuk memproses tugas sementara `GetActivityTask` diblokir.

Untuk menghindari masalah latensi seperti ini saat melakukan polling untuk tugas:

- Terapkan poller Anda sebagai utas terpisah dari tugas dalam pelaksanaan pekerja aktivitas Anda.
- Miliki setidaknya 100 polling terbuka per aktivitas ARN di setiap titik waktu.

> **Note**
>
> Penskalaan ke 100 polling terbuka per ARN bisa mahal. Misalnya, 100 polling fungsi Lambda per ARN 100 kali lebih mahal daripada memiliki fungsi Lambda tunggal dengan 100 utas polling. Untuk mengurangi latensi dan meminimalkan biaya, gunakan bahasa yang memiliki asinkron I/O, dan terapkan beberapa utas polling per pekerja. Untuk pekerja aktivitas contoh tempat utas poller terpisah dari utas tugas, lihat Contoh Activity Worker di Ruby.

Untuk informasi selengkapnya tentang aktivitas dan pekerja aktivitas lihat Aktivitas.

Memilih Alur Kerja Standar atau Ekspres

AWS Step Functions menawarkan Alur Kerja Standar sebagai tipe alur kerja default, dengan opsi untuk memilih Alur Kerja Ekspres.

Anda dapat memilih Alur Kerja Standar bila Anda memerlukan alur kerja yang berjalan lama, tahan lama, dan dapat diaudit, atau Alur Kerja Ekspres untuk beban kerja pemrosesan peristiwa bervolume tinggi. Eksekusi mesin status Anda akan berperilaku berbeda, tergantung pada `Type` yang Anda pilih. Parameter `Type` yang Anda pilih tidak dapat diubah setelah mesin status Anda telah dibuat.

- Untuk informasi rinci tentang perbedaan antara Alur Kerja Standar dan Ekspres, lihatAlur Kerja Standar vs Ekspres.
- Untuk informasi tentang mengoptimalkan biaya saat membangun alur kerja tanpa server menggunakan Step Functions, lihat. Optimalisasi biaya menggunakan Alur Kerja Ekspres

Pembatasan ukuran kebijakan sumber daya Amazon CloudWatch Logs

CloudWatch Kebijakan sumber daya log dibatasi hingga 5120 karakter. Ketika CloudWatch Log mendeteksi bahwa kebijakan mendekati batas ukuran ini, secara otomatis mengaktifkan grup log yang memulai`/aws/vendedlogs/`.

Saat Anda membuat mesin status dengan logging diaktifkan, Step Functions harus memperbarui kebijakan sumber daya CloudWatch Log Anda dengan grup log yang Anda tentukan. Agar tidak mencapai batas ukuran kebijakan sumber daya CloudWatch Log, awali nama grup CloudWatch log Log Anda dengan`/aws/vendedlogs/`. Ketika Anda membuat grup log di konsol Step Functions, nama grup log diawali dengan `/aws/vendedlogs/states`. Untuk informasi selengkapnya, lihat Mengaktifkan Pencatatan dari Layanan AWS Tertentu.

Jika Anda tidak dapat mengakses CloudWatch Log, lihat Unable to access the CloudWatch Logs untuk informasi.

Menggunakan AWS Step Functions dengan layanan lain

Pelajari cara mengoordinasikan API pihak ketiga lainnya Layanan AWS atau memanggil API pihak ketiga. AWS Step Functions

Topik

- Hubungi AWS layanan lain
- AWS Integrasi layanan SDK
- Integrasi yang dioptimalkan untuk Step Functions
- Panggil API pihak ketiga
- Pola integrasi layanan
- Meneruskan parameter ke API layanan
- Ubah log untuk integrasi AWS SDK yang didukung

Hubungi AWS layanan lain

Dengan integrasi AWS layanan, Anda dapat memanggil tindakan API dan mengoordinasikan eksekusi langsung dari alur kerja Anda. Anda dapat menggunakan integrasi AWS SDK Fungsi Langkah untuk memanggil salah satu dari lebih dari dua ratus AWS layanan langsung dari mesin status Anda, memberi Anda akses ke lebih dari sembilan ribu tindakan API. Atau Anda dapat menggunakan integrasi Fungsi Langkah yang Dioptimalkan, yang masing-masing telah disesuaikan untuk menyediakan fungsionalitas khusus untuk alur kerja Anda. Beberapa tindakan API tersedia di kedua jenis integrasi. Jika memungkinkan, sebaiknya gunakan integrasi yang Dioptimalkan.

Anda mengoordinasikan layanan ini secara langsung dari status `Task` dalam Amazon States Language. Misalnya, menggunakan Step Functions, Anda dapat memanggil layanan lain untuk:

- Memanggil AWS Lambda fungsi.
- Jalankan AWS Batch pekerjaan dan kemudian lakukan tindakan yang berbeda berdasarkan hasil.
- Memasukkan atau mendapatkan item dari Amazon DynamoDB.
- Menjalankan tugas Amazon Elastic Container Service (Amazon ECS) dan menunggu hingga selesai.
- Menerbitkan topik di Amazon Simple Notification Service (Amazon SNS).
- Mengirim pesan di Amazon Simple Queue Service (Amazon SQS).

- Kelola pekerjaan untuk AWS Glue atau Amazon SageMaker.
- Membangun alur kerja untuk melaksanakan tugas Amazon EMR.
- Luncurkan eksekusi AWS Step Functions alur kerja.

Integrasi yang dioptimalkan

Integrasi yang dioptimalkan telah disesuaikan dengan Step Functions untuk menyediakan fungsionalitas khusus untuk konteks alur kerja. Misalnya, Lambda `Invoke` mengonversi output API-nya dari JSON yang lolos ke objek JSON. AWS Batch`SubmitJob`memungkinkan Anda menjeda eksekusi sampai pekerjaan selesai. Set pertama integrasi yang dioptimalkan dirilis pada tahun 2018, dan sekarang ada lebih dari lima puluh API.

AWS Integrasi SDK

AWS Integrasi SDK bekerja persis seperti panggilan API standar menggunakan SDK AWS . Mereka menyediakan kemampuan untuk memanggil lebih dari sembilan ribu API di lebih dari dua ratus AWS layanan langsung dari definisi mesin negara Anda.

Dukungan pola integrasi

Alur Kerja Standar dan Alur Kerja Ekspres mendukung integrasi yang sama tetapi bukan pola integrasi yang sama.

- Dukungan pola integrasi yang dioptimalkan berbeda untuk setiap integrasi.
- Alur Kerja Ekspres tidak mendukung Run a Job (.sync) atau Wait for Callback (.waitForTaskToken).
- Untuk informasi selengkapnya, lihat Alur Kerja Standar vs Ekspres.

Standard Workflows

Integrasi layanan yang didukung

	Layanan	Permintaan Respon	Jalankan Job (`.sync`)	Tunggu Callback () `.waitForTaskToken`
Integrasi yang dioptimalkan	Amazon API Gateway	✓		✓
	Amazon Athena	✓	✓	
	AWS Batch	✓	✓	
	Amazon Bedrock	✓	✓	✓
	AWS CodeBuild	✓	✓	
	Amazon DynamoDB	✓		
	Amazon ECS/Fargate	✓	✓	✓
	Amazon EKS	✓	✓	✓
	Amazon EMR	✓	✓	
	Amazon EMR on EKS	✓	✓	
	Amazon EMR Serverless	✓	✓	
	Amazon EventBridge	✓		✓
	AWS Glue	✓	✓	
	AWS Glue DataBrew	✓	✓	
	AWS Lambda	✓		✓
	AWS Elemental MediaConvert	✓	✓	

	Layanan	Permintaan Respon	Jalankan Job (`.sync`)	Tunggu Callback () `.waitForTaskToken`
	Amazon SageMaker	✓	✓	
	Amazon SNS	✓		✓
	Amazon SQS	✓		✓
	AWS Step Functions	✓	✓	✓
AWS Integrasi SDK	Lebih dari dua ratus	✓		✓

Express Workflows

Integrasi layanan yang didukung

	Layanan	Permintaan Respon	Jalankan Job (`.sync`)	Tunggu Callback () `.waitForTaskToken`
Integrasi yang dioptimalkan	Amazon API Gateway	✓		
	Amazon Athena	✓		
	AWS Batch	✓		
	Amazon Bedrock	✓		
	AWS CodeBuild	✓		
	Amazon DynamoDB	✓		

	Layanan	Permintaan Respon	Jalankan Job (`.sync`)	Tunggu Callback () `.waitForTaskToken`
	Amazon ECS/Fargate	✓		
	Amazon EKS	✓		
	Amazon EMR	✓		
	Amazon EMR on EKS	✓		
	Amazon EMR Serverless	✓		
	Amazon EventBridge	✓		
	AWS Glue	✓		
	AWS Glue DataBrew	✓		
	AWS Lambda	✓		
	AWS Elemental MediaConvert	✓		
	Amazon SageMaker	✓		
	Amazon SNS	✓		
	Amazon SQS	✓		
	AWS Step Functions	✓		
AWS Integrasi SDK	Lebih dari dua ratus	✓		

Akses lintas akun

Step Functions menyediakan akses lintas akun ke sumber daya yang dikonfigurasi Akun AWS di berbagai alur kerja Anda. Dengan menggunakan integrasi layanan Step Functions, Anda dapat memanggil AWS sumber daya lintas akun apa pun meskipun Layanan AWS tidak mendukung kebijakan berbasis sumber daya atau panggilan lintas akun.

Untuk informasi selengkapnya, lihat Mengakses sumber daya di tempat lain Akun AWS di alur kerja Anda.

AWS Integrasi layanan SDK

AWS Step Functions terintegrasi dengan Layanan AWS, memungkinkan Anda memanggil tindakan API setiap layanan dari alur kerja Anda. Anda dapat menggunakan integrasi AWS SDK Fungsi Langkah untuk memanggil hampir Layanan AWS semua tindakan API dari mesin status Anda. Anda juga dapat menggunakan integrasi Fungsi Langkah yang Dioptimalkan, yang masing-masing telah disesuaikan untuk menyediakan fungsionalitas khusus untuk alur kerja Anda.

Beberapa layanan atau SDK mungkin tidak tersedia sebagai integrasi AWS SDK, baik sementara maupun permanen. Layanan yang baru dirilis mungkin tidak memiliki interaksi SDK yang tersedia hingga pembaruan nanti. Beberapa layanan memerlukan konfigurasi yang disesuaikan, seperti menentukan titik akhir khusus pelanggan, yang mungkin lebih cocok untuk integrasi yang dioptimalkan. SDK lain tidak cocok untuk digunakan dalam alur kerja, seperti untuk streaming audio atau video. Akhirnya, beberapa layanan dapat ditahan sampai melewati validasi internal tertentu yang dilakukan oleh Step Functions.

> **Tip**
>
> Untuk menerapkan contoh alur kerja yang menggunakan integrasi AWS SDK ke Anda Akun AWS, lihat Modul 9 - Integrasi layanan AWS SDK di The Workshop. AWS Step Functions

Topik

- Menggunakan AWS integrasi layanan SDK
- Integrasi layanan AWS SDK yang didukung
- Tindakan API yang tidak didukung untuk layanan yang didukung

- Integrasi layanan SDK yang tidak digunakan AWS lagi

Menggunakan AWS integrasi layanan SDK

Untuk menggunakan integrasi AWS SDK, Anda menentukan nama layanan dan panggilan API dan, secara opsional, pola integrasi layanan.

> **Note**
>
> - Parameter dalam Step Functions dinyatakan dalam PascalCase, bahkan jika API layanan asli ada di camelCase. Misalnya, Anda dapat menggunakan tindakan Step Functions API `startSyncExecution` dan menentukan parameternya sebagai`StateMachineArn`.
> - Untuk tindakan API yang menerima parameter yang disebutkan, seperti tindakan `DescribeLaunchTemplateVersions` API untuk Amazon EC2, tentukan versi jamak dari nama parameter. Misalnya, tentukan `Filters` `Filter.N` parameter tindakan `DescribeLaunchTemplateVersions` API.

Anda dapat memanggil layanan AWS SDK langsung dari Amazon States Language di `Resource` bidang status tugas. Untuk melakukan ini, gunakan sintaks berikut:

`arn:aws:states:::aws-sdk:`*`serviceName`*`:`*`apiAction`*`.`*`[serviceIntegrationPattern]`*

Misalnya, untuk Amazon EC2, Anda mungkin menggunakannya. `arn:aws:states:::aws-sdk:ec2:describeInstances` Ini akan mengembalikan output seperti yang ditentukan untuk panggilan API DescribeInstances Amazon EC2.

Jika integrasi AWS SDK mengalami kesalahan, bidang Kesalahan yang dihasilkan akan terdiri dari nama layanan dan nama kesalahan, dipisahkan oleh karakter periode:. *`ServiceName`*`.`*`ErrorName`* Baik nama layanan dan nama kesalahan dalam kasus Pascal. Anda juga dapat melihat nama layanan, dalam huruf kecil, di bidang Sumber Daya status tugas. Anda dapat menemukan nama kesalahan potensial dalam dokumentasi referensi API layanan target.

Misalnya, Anda mungkin menggunakan integrasi `arn:aws:states:::aws-sdk:acmpca:deleteCertificateAuthority` AWS SDK. Referensi AWS Private Certificate AuthorityDeleteCertificateAuthority API menunjukkan bahwa tindakan API dapat menghasilkan`ResourceNotFoundException`, misalnya. Untuk menangani kesalahan ini, Anda

akan menentukan Kesalahan `AcmPca.ResourceNotFoundException` dalam Retrier atau Penangkap status Tugas Anda. Untuk informasi selengkapnya tentang penanganan kesalahan di Step Functions, lihatPenanganan kesalahan dalam Step Functions.

Step Functions tidak dapat membuat kebijakan IAM secara otomatis untuk integrasi AWS SDK. Setelah membuat mesin status, Anda harus menavigasi ke konsol IAM dan mengonfigurasi kebijakan peran Anda. Untuk informasi selengkapnya, lihat Kebijakan IAM untuk layanan terintegrasi.

Lihat Kumpulkan info bucket Amazon S3 menggunakan integrasi layanan AWS SDK tutorial untuk contoh cara menggunakan integrasi AWS SDK.

Integrasi layanan AWS SDK yang didukung

Tabel berikut mencantumkan integrasi layanan AWS SDK yang didukung oleh Step Functions. Kolom sumber daya *`Task`* status mencantumkan sintaks untuk memanggil tindakan API tertentu saat menggunakan integrasi untuk layanan yang ditentukan dalam kolom Nama layanan di sebelah kiri. Kolom Tanggal yang didukung memberikan informasi tentang tanggal di mana integrasi layanan didukung. Selain itu, kolom awalan Exception di sebelah kanan, mencantumkan awalan pengecualian untuk setiap integrasi layanan. Awalan pengecualian ini hadir dalam pengecualian yang dihasilkan saat Anda salah melakukan AWS integrasi layanan SDK dengan Step Functions.

> **Note**
>
> - Layanan yang ditandai dengan*** memiliki tindakan API yang tidak didukung oleh Step Functions saat ini. Untuk informasi tentang tindakan yang tidak didukung untuk layanan, lihat Tindakan API yang tidak didukung untuk layanan yang didukung tabel.
> - Untuk informasi tentang pembaruan yang dilakukan pada setiap peluncuran guna memperluas dukungan integrasi AWS SDK, lihat. Ubah log untuk integrasi AWS SDK yang didukung

> **Important**
>
> Dukungan tindakan API dirilis pada irama triwulanan. Pembaruan untuk tindakan yang sudah didukung, seperti parameter baru, mungkin tidak segera tersedia.

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
AWS AppFabric	`arn:aws:states:::aws-sdk:appfabric: [apiAction]`	Januari 18, 2024	AppFabric
B2B Data Interchange	`arn:aws:states:::aws-sdk:b2bi: [apiAction]`	Januari 18, 2024	B2Bi
Ekspor Data AWS	`arn:aws:states:::aws-sdk:bcmdataexports: [apiAction]`	Januari 18, 2024	BcmDataExports
Amazon Bedrock	`arn:aws:states:::aws-sdk:bedrock: [apiAction]`	Januari 18, 2024	Bedrock
Amazon Bedrock Agents	`arn:aws:states:::aws-sdk:bedrockagent: [apiAction]`	Januari 18, 2024	BedrockAgent
Amazon Bedrock Runtime Agents	`arn:aws:states:::aws-sdk:bedrockagentruntime: [apiAction]`	Januari 18, 2024	BedrockAgentRuntime
Amazon Bedrock Runtime	`arn:aws:states:::aws-sdk:bedrockruntime: [apiAction]`	Januari 18, 2024	BedrockRuntime
AWS Clean Rooms	`arn:aws:states:::aws-sdk:cleanroomsml: [apiAction]`	Januari 18, 2024	CleanRoomsML
Amazon CloudFront KeyValueStore	`arn:aws:states:::aws-sdk:cl`	Januari 18, 2024	CloudFrontKeyValueStore

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
	`oudfrontkeyvaluestore: [apiAction]`		
CodeGuru Security	`arn:aws:states:::aws-sdk:codegurusecurity: [apiAction]`	Januari 18, 2024	CodeGuruSecurity
Hub Optimisasi Biaya AWS	`arn:aws:states:::aws-sdk:costoptimizationhub: [apiAction]`	Januari 18, 2024	CostOptimizationHub
Amazon DataZone	`arn:aws:states:::aws-sdk:datazone: [apiAction]`	Januari 18, 2024	DataZone
Amazon EKS Auth	`arn:aws:states:::aws-sdk:eksauth: [apiAction]`	Januari 18, 2024	EksAuth
Resolusi Entitas AWS	`arn:aws:states:::aws-sdk:entityresolution: [apiAction]`	Januari 18, 2024	EntityResolution
AWS Tingkat Gratis	`arn:aws:states:::aws-sdk:freetier: [apiAction]`	Januari 18, 2024	FreeTier
Amazon Inspector Scan	`arn:aws:states:::aws-sdk:inspectorscan: [apiAction]`	Januari 18, 2024	InspectorScan
AWS Launch Wizard	`arn:aws:states:::aws-sdk:launchwizard: [apiAction]`	Januari 18, 2024	LaunchWizard

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
Amazon Managed Blockchain Query	`arn:aws:states:::aws-sdk:managedblockchainquery: [apiAction]`	Januari 18, 2024	ManagedBlockchainQuery
AWS Elemental MediaPackage V2	`arn:aws:states:::aws-sdk:mediapackagev2: [apiAction]`	Januari 18, 2024	MediaPackageV2
AWS HealthImaging	`arn:aws:states:::aws-sdk:medicalimaging: [apiAction]`	Januari 18, 2024	MedicalImaging
Network Manager	`arn:aws:states:::aws-sdk:networkmanager: [apiAction]`	Januari 18, 2024	NetworkManager
AWS Payment Cryptography	`arn:aws:states:::aws-sdk:paymentcryptography: [apiAction]`	Januari 18, 2024	PaymentCryptography
AWS Payment Cryptography Data	`arn:aws:states:::aws-sdk:paymentcryptographydata: [apiAction]`	Januari 18, 2024	PaymentCryptographyData
AWS Private CA Connector for Active Directory	`arn:aws:states:::aws-sdk:pcaconnectorad: [apiAction]`	Januari 18, 2024	PcaConnectorAd
Amazon Q Business	`arn:aws:states:::aws-sdk:qbusiness: [apiAction]`	Januari 18, 2024	QBusiness

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
Amazon Q Connect	`arn:aws:states:::aws-sdk:qconnect: [apiAction]`	Januari 18, 2024	QConnect
AWS re:Post	`arn:aws:states:::aws-sdk:repostspace: [apiAction]`	Januari 18, 2024	Repostspace
Amazon Timestream Query	`arn:aws:states:::aws-sdk:timestreamquery: [apiAction]`	Januari 18, 2024	TimestreamQuery
Amazon Timestream Write	`arn:aws:states:::aws-sdk:timestreamwrite: [apiAction]`	Januari 18, 2024	TimestreamWrite
Trusted Advisor	`arn:aws:states:::aws-sdk:trustedadvisor: [apiAction]`	Januari 18, 2024	TrustedAdvisor
Verified Permissions	`arn:aws:states:::aws-sdk:verifiedpermissions: [apiAction]`	Januari 18, 2024	VerifiedPermissions
Amazon WorkSpaces Thin Client	`arn:aws:states:::aws-sdk:workspacesthinclient: [apiAction]`	Januari 18, 2024	WorkSpacesThinClient
AWS CloudTrail Data	`arn:aws:states:::aws-sdk:cloudtraildata: [apiAction]`	Juni 16, 2023	CloudTrailData

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
Amazon CloudWatch Internet Monitor	`arn:aws:states:::aws-sdk:internetmonitor: [apiAction]`	Juni 16, 2023	InternetMonitor
Amazon Interactive Video Service RealTime	`arn:aws:states:::aws-sdk:ivsrealtime: [apiAction]`	Juni 16, 2023	IvsRealTime
AWS IoT TwinMaker	`arn:aws:states:::aws-sdk:iottwinmaker: [apiAction]`	Juni 16, 2023	IoTTwinMaker
Amazon OpenSearch Ingestion	`arn:aws:states:::aws-sdk:osis: [apiAction]`	Juni 16, 2023	Osis
AWS Telco Network Builder	`arn:aws:states:::aws-sdk: tnb:[apiAction]`	Juni 16, 2023	Tnb
Amazon VPC Lattice	`arn:aws:states:::aws-sdk:vpclattice: [apiAction]`	Juni 16, 2023	VpcLattice
Amazon Chime Media Pipelines	`arn:aws:states:::aws-sdk:chimesdkmediapipelines: [apiAction]`	17 Februari 2023	ChimeSdkMediaPipelines
Amazon Chime Voice	`arn:aws:states:::aws-sdk:chimesdkvoice: [apiAction]`	17 Februari 2023	ChimeSdkVoice

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
Amazon CodeCatalyst	`arn:aws:states:::aws-sdk:codecatalyst: [apiAction]`	17 Februari 2023	CodeCatalyst
Amazon Connect Cases	`arn:aws:states:::aws-sdk:connectcases: [apiAction]`	17 Februari 2023	ConnectCases
Amazon DocumentDB Elastic Clusters	`arn:aws:states:::aws-sdk:docdbelastic: [apiAction]`	17 Februari 2023	DocDbElastic
Amazon EMR Serverless	`arn:aws:states:::aws-sdk:emrserverless: [apiAction]`	17 Februari 2023	EmrServerless
Amazon IVS Chat	`arn:aws:states:::aws-sdk:ivs: [apiAction]`	17 Februari 2023	Ivs
Amazon Kendra Intelligent Ranking	`arn:aws:states:::aws-sdk:kendrarankingng: [apiAction]`	17 Februari 2023	KendraRanking
AWS HealthOmics	`arn:aws:states:::aws-sdk:omics: [apiAction]`	17 Februari 2023	Omics
Amazon Redshift Serverless	`arn:aws:states:::aws-sdk:redshiftserverless: [apiAction]`	17 Februari 2023	RedshiftServerless
Amazon Security Lake	`arn:aws:states:::aws-sdk:securitylake: [apiAction]`	17 Februari 2023	SecurityLake

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
AWS Backup Storage	`arn:aws:states:::aws-sdk:backupstorage: [apiAction]`	17 Februari 2023	BackupStorage
AWS Clean Rooms	`arn:aws:states:::aws-sdk:cleanrooms: [apiAction]`	17 Februari 2023	CleanRooms
AWS Control Tower	`arn:aws:states:::aws-sdk:controltower: [apiAction]`	17 Februari 2023	ControlTower
AWS Health	`arn:aws:states:::aws-sdk:health: [apiAction]`	17 Februari 2023	Health
AWS IoT FleetWise	`arn:aws:states:::aws-sdk:iotfleetwise: [apiAction]`	17 Februari 2023	IotFleetWise
AWS IoT RoboRunner	`arn:aws:states:::aws-sdk:iotroborunner: [apiAction]`	17 Februari 2023	IotRoboRunner
AWS Mainframe Modernization	`arn:aws:states:::aws-sdk:m2: [apiAction]`	17 Februari 2023	M2
Orkestrator AWS Migration Hub	`arn:aws:states:::aws-sdk:migrationhuborchestrator: [apiAction]`	17 Februari 2023	Migration HubOrchestrator

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
AWS Private 5G	`arn:aws:states:::aws-sdk:privatenetworks: [apiAction]`	17 Februari 2023	PrivateNetworks
Penjelajah Sumber Daya AWS	`arn:aws:states:::aws-sdk:resourceexplorer2: [apiAction]`	17 Februari 2023	ResourceExplorer2
AWS SimSpace Weaver	`arn:aws:states:::aws-sdk:simspaceweaver: [apiAction]`	17 Februari 2023	SimSpaceWeaver
AWS Support App	`arn:aws:states:::aws-sdk:supportapp: [apiAction]`	17 Februari 2023	SupportApp
CloudWatch Observability Access Manager	`arn:aws:states:::aws-sdk:oam: [apiAction]`	17 Februari 2023	Oam
EventBridge Pipes	`arn:aws:states:::aws-sdk:pipes: [apiAction]`	17 Februari 2023	Pipes
EventBridge Scheduler	`arn:aws:states:::aws-sdk:scheduler: [apiAction]`	17 Februari 2023	Scheduler
IAM Roles Anywhere	`arn:aws:states:::aws-sdk:rolesanywhere: [apiAction]`	17 Februari 2023	RolesAnywhere

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
Kinesis Video WebRTC Storage	`arn:aws:states:::aws-sdk:kinesisvideowebrtcstorage: [apiAction]`	17 Februari 2023	KinesisVideoWebRtcStorage
License Manager Linux Subscriptions	`arn:aws:states:::aws-sdk:licensemanagerlinuxsubscriptions: [apiAction]`	17 Februari 2023	LicenseManagerLinuxSubscriptions
License Manager User Subscriptions	`arn:aws:states:::aws-sdk:licensemanagerusersubscriptions: [apiAction]`	17 Februari 2023	LicenseManagerUserSubscriptions
OpenSearch Serverless	`arn:aws:states:::aws-sdk:opensearchserverless: [apiAction]`	17 Februari 2023	OpenSearchServerless
Route 53 ARC Zonal Shift	`arn:aws:states:::aws-sdk:arczonalshift: [apiAction]`	17 Februari 2023	ArcZonalShift
SageMaker Geospatial	`arn:aws:states:::aws-sdk:sagemakergeospatial: [apiAction]`	17 Februari 2023	SageMaker Geospatial
SageMaker Metrics	`arn:aws:states:::aws-sdk:sagemakermetrics: [apiAction]`	17 Februari 2023	SageMakerMetrics

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
Systems Manager for SAP	`arn:aws:states:::aws-sdk:ssmsap: [apiAction]`	17 Februari 2023	SsmSap
AWS Account Management	`arn:aws:states:::aws-sdk:account: [apiAction]`	19 April 2022	Account
AWS Amplify	`arn:aws:states:::aws-sdk:amplify: [apiAction]`	30 September 2021	Amplify
AWS App Mesh	`arn:aws:states:::aws-sdk:appmesh: [apiAction]`	30 September 2021	AppMesh
AWS App Runner	`arn:aws:states:::aws-sdk:apprunner: [apiAction]`	30 September 2021	AppRunner
AWS AppConfig	`arn:aws:states:::aws-sdk:appconfig: [apiAction]`	30 September 2021	AppConfig
AWS AppConfig Data	`arn:aws:states:::aws-sdk:appconfigdata: [apiAction]`	19 April 2022	AppConfigData
AWS AppSync	`arn:aws:states:::aws-sdk:appsync: [apiAction]`	30 September 2021	AppSync

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
AWS Application Discovery Service	`arn:aws:states:::aws-sdk:applicationdiscovery:` *`[apiAction]`* ***	30 September 2021	ApplicationDiscovery
AWS Application Migration Service	`arn:aws:states:::aws-sdk:mgn:` *`[apiAction]`*	30 September 2021	Mgn
AWS Audit Manager	`arn:aws:states:::aws-sdk:auditmanager:` *`[apiAction]`*	30 September 2021	AuditManager
AWS Auto Scaling Plans	`arn:aws:states:::aws-sdk:autoscalingplans:` *`[apiAction]`*	30 September 2021	AutoScalingPlans
AWS Backup	`arn:aws:states:::aws-sdk:backup:` *`[apiAction]`*	30 September 2021	Backup
AWS Backup gateway	`arn:aws:states:::aws-sdk:backupgateway:` *`[apiAction]`*	19 April 2022	BackupGateway
AWS Batch	`arn:aws:states:::aws-sdk:batch:` *`[apiAction]`*	30 September 2021	Batch
AWS Billing Conductor	`arn:aws:states:::aws-sdk:billingconductor:` *`[apiAction]`*	26 Juli 2022	Billingconductor

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
AWS Budgets	`arn:aws:states:::aws-sdk:budgets: [apiAction]`	30 September 2021	Budgets
AWS Certificate Manager	`arn:aws:states:::aws-sdk:acm: [apiAction]`	30 September 2021	Acm
AWS Private Certificate Authority	`arn:aws:states:::aws-sdk:acmpca: [apiAction]`	30 September 2021	AcmPca
AWS Cloud Map	`arn:aws:states:::aws-sdk:servicediscovery: [apiAction]`	30 September 2021	ServiceDiscovery
AWS Cloud9	`arn:aws:states:::aws-sdk:cloud9: [apiAction]`	30 September 2021	Cloud9
AWS CloudFormation	`arn:aws:states:::aws-sdk:cloudformation: [apiAction]`	30 September 2021	CloudFormation
AWS CloudHSM	`arn:aws:states:::aws-sdk:cloudhsm: [apiAction]`	30 September 2021	CloudHsm
AWS CloudHSM	`arn:aws:states:::aws-sdk:cloudhsmv2: [apiAction]`	30 September 2021	CloudHsmV2
AWS CloudTrail	`arn:aws:states:::aws-sdk:cloudtrail: [apiAction]`	30 September 2021	CloudTrail

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
AWS Cloud Control	`arn:aws:states:::aws-sdk:cloudcontrol: [apiAction]`	19 April 2022	CloudControl
AWS CodeBuild	`arn:aws:states:::aws-sdk:codebuild: [apiAction]`	30 September 2021	CodeBuild
AWS CodeCommit	`arn:aws:states:::aws-sdk:codecommit: [apiAction]`	30 September 2021	CodeCommit
AWS CodeDeploy	`arn:aws:states:::aws-sdk:codedeploy: [apiAction]` ***	30 September 2021	CodeDeploy
AWS CodePipeline	`arn:aws:states:::aws-sdk:codepipeline: [apiAction]`	30 September 2021	CodePipeline
AWS CodeStar	`arn:aws:states:::aws-sdk:codestar: [apiAction]`	30 September 2021	CodeStar
AWS CodeStar	`arn:aws:states:::aws-sdk:codestarnotifications: [apiAction]`	30 September 2021	CodestarNotifications
AWS CodeStar	`arn:aws:states:::aws-sdk:codestarconnections: [apiAction]`	30 September 2021	CodeStarConnections

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
AWS Compute Optimizer	`arn:aws:states:::aws-sdk:computeoptimizer: [apiAction]`	30 September 2021	ComputeOptimizer
AWS Config	`arn:aws:states:::aws-sdk:config: [apiAction]`	30 September 2021	Config
AWS Cost Explorer Service	`arn:aws:states:::aws-sdk:costexplorer: [apiAction]`	30 September 2021	CostExplorer
AWS Cost and Usage Report	`arn:aws:states:::aws-sdk:costandusagereport: [apiAction]`	30 September 2021	CostAndUsageReport
AWS Data Exchange	`arn:aws:states:::aws-sdk:dataexchange: [apiAction]`	30 September 2021	DataExchange
AWS Data Pipeline	`arn:aws:states:::aws-sdk:datapipeline: [apiAction]`	30 September 2021	DataPipeline
AWS DataSync	`arn:aws:states:::aws-sdk:datasync: [apiAction]`	30 September 2021	DataSync
AWS Database Migration Service	`arn:aws:states:::aws-sdk:databasemigration: [apiAction]`	30 September 2021	DatabaseMigration
AWS Device Farm	`arn:aws:states:::aws-sdk:devicefarm: [apiAction]`	30 September 2021	DeviceFarm

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
AWS Direct Connect	`arn:aws:states:::aws-sdk:directconnect: [apiAction]` *** —	30 September 2021	DirectConnect
AWS Directory Service	`arn:aws:states:::aws-sdk:directory: [apiAction]`	30 September 2021	Directory
AWS EC2 Instance Connect	`arn:aws:states:::aws-sdk:ec2instanceconnect: [apiAction]`	30 September 2021	Ec2InstanceConnect
AWS Elastic Beanstalk	`arn:aws:states:::aws-sdk:elasticbeanstalk: [apiAction]`	30 September 2021	ElasticBeanstalk
AWS Elemental MediaLive	`arn:aws:states:::aws-sdk:medialive: [apiAction]`	30 September 2021	MediaLive
AWS Elemental MediaPackage	`arn:aws:states:::aws-sdk:mediapackage: [apiAction]` *** —	30 September 2021	MediaPackage
AWS Elemental MediaPackage VOD	`arn:aws:states:::aws-sdk:mediapackagevod: [apiAction]`	30 September 2021	MediaPackageVod
AWS Elemental MediaStore	`arn:aws:states:::aws-sdk:mediastore: [apiAction]`	30 September 2021	MediaStore
AWS Fault Injection Service	`arn:aws:states:::aws-sdk:fis: [apiAction]`	30 September 2021	Fis

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
AWS Firewall Manager	`arn:aws:states:::aws-sdk:fms:` *`[apiAction]`*	30 September 2021	Fms
AWS Glue	`arn:aws:states:::aws-sdk:glue:` *`[apiAction]`*	30 September 2021	Glue
AWS Glue DataBrew	`arn:aws:states:::aws-sdk:databrew:` *`[apiAction]`*	30 September 2021	DataBrew
AWS IoT Greengrass	`arn:aws:states:::aws-sdk:greengrass:` *`[apiAction]`*	30 September 2021	Greengrass
AWS Ground Station	`arn:aws:states:::aws-sdk:groundstation:` *`[apiAction]`*	30 September 2021	GroundStation
AWS Identity and Access Management	`arn:aws:states:::aws-sdk:iam:` *`[apiAction]`*	30 September 2021	Iam
AWS IoT	`arn:aws:states:::aws-sdk:iot:` *`[apiAction]`* ***	30 September 2021	Iot
AWS IoT 1-Click	`arn:aws:states:::aws-sdk:iot1clickprojects:` *`[apiAction]`*	30 September 2021	Iot1ClickProjects
AWS IoT Analytics	`arn:aws:states:::aws-sdk:iotanalytics:` *`[apiAction]`*	30 September 2021	IoTAnalytics

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
AWS IoT Core Device Advisor	`arn:aws:states:::aws-sdk:iotdeviceadvisor: [apiAction]` *** —	30 September 2021	IotDeviceAdvisor
AWS IoT Events	`arn:aws:states:::aws-sdk:iotevents: [apiAction]`	30 September 2021	IotEvents
Data AWS IoT Events	`arn:aws:states:::aws-sdk:ioteventsdata: [apiAction]`	30 September 2021	IotEventsData
AWS IoT Fleet Hub	`arn:aws:states:::aws-sdk:iotfleethub: [apiAction]`	30 September 2021	IoTFleetHub
AWS IoT Greengrass Version 2	`arn:aws:states:::aws-sdk:greengrassv2: [apiAction]`	30 September 2021	GreengrassV2
Data tugas AWS IoT Plane	`arn:aws:states:::aws-sdk:iotjobsdataplane: [apiAction]`	30 September 2021	IotJobsDataPlane
AWS IoT Secure Tunneling	`arn:aws:states:::aws-sdk:iotsecuretunneling: [apiAction]`	30 September 2021	IoTSecure Tunneling
AWS IoT SiteWise	`arn:aws:states:::aws-sdk:iotsitewise: [apiAction]`	30 September 2021	IoTSiteWise
AWS IoT Wireless	`arn:aws:states:::aws-sdk:iotwireless: [apiAction]`	30 September 2021	IotWireless

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
AWS Key Management Service	`arn:aws:states:::aws-sdk:kms:` *`[apiAction]`*	30 September 2021	Kms
AWS Lake Formation	`arn:aws:states:::aws-sdk:lakeformation:` *`[apiAction]`*	30 September 2021	LakeFormation
AWS Lambda	`arn:aws:states:::aws-sdk:lambda:` *`[apiAction]`* ***	30 September 2021	Lambda
AWS License Manager	`arn:aws:states:::aws-sdk:licensemanager:` *`[apiAction]`*	30 September 2021	LicenseManager
AWS Marketplace	`arn:aws:states:::aws-sdk:marketplacecatalog:` *`[apiAction]`*	30 September 2021	MarketplaceCatalog
AWS Marketplace Commerce Analytics	`arn:aws:states:::aws-sdk:marketplacecommerceanalytics:` *`[apiAction]`*	30 September 2021	MarketplaceCommerceAnalytics
AWS Marketplace Entitlement Service	`arn:aws:states:::aws-sdk:marketplaceentitlement:` *`[apiAction]`*	30 September 2021	MarketplaceEntitlement
AWS Elemental MediaTailor	`arn:aws:states:::aws-sdk:mediatailor:` *`[apiAction]`*	30 September 2021	MediaTailor

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
AWS Migration Hub	`arn:aws:states:::aws-sdk:migrationhub: [apiAction]`	30 September 2021	MigrationHub
AWS Migration Hub Config	`arn:aws:states:::aws-sdk:migrationhubconfig: [apiAction]`	30 September 2021	MigrationHubConfig
Rekomendasi Strategi AWS Migration Hub	`arn:aws:states:::aws-sdk:migrationhubstrategy: [apiAction]`	19 April 2022	MigrationHubStrategy
AWS Mobile	`arn:aws:states:::aws-sdk:mobile: [apiAction]`	30 September 2021	
AWS Network Firewall	`arn:aws:states:::aws-sdk:networkfirewall: [apiAction]`	30 September 2021	NetworkFirewall
AWS OpsWorks	`arn:aws:states:::aws-sdk:opsworks: [apiAction]`	30 September 2021	OpsWorks
AWS OpsWorks CM	`arn:aws:states:::aws-sdk:opsworkscm: [apiAction]`	30 September 2021	OpsWorksCm
AWS Organizations	`arn:aws:states:::aws-sdk:organizations: [apiAction]`	30 September 2021	Organizations

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
AWS Outposts	`arn:aws:states:::aws-sdk:outposts: [apiAction]`	30 September 2021	Outposts
AWS Panorama	`arn:aws:states:::aws-sdk:panorama: [apiAction]`	19 April 2022	Panorama
Amazon Relational Database Service Performance Insights	`arn:aws:states:::aws-sdk:pi: [apiAction]`	30 September 2021	Pi
Daftar Harga AWS	`arn:aws:states:::aws-sdk:pricing: [apiAction]`	30 September 2021	Pricing
Amazon Relational Database Service	`arn:aws:states:::aws-sdk:rdsdata: [apiAction]` ***	30 September 2021	RdsData
AWS Resilience Hub	`arn:aws:states:::aws-sdk:resiliencehub: [apiAction]`	19 April 2022	Resiliencehub
AWS Resource Access Manager	`arn:aws:states:::aws-sdk:ram: [apiAction]`	30 September 2021	Ram
AWS Resource Groups	`arn:aws:states:::aws-sdk:resourcegroups: [apiAction]`	30 September 2021	ResourceGroups

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
AWS Resource Groups Tagging API	`arn:aws:states:::aws-sdk:resourcegroupstaggingapi: [apiAction]`	30 September 2021	ResourceGroupsTaggingApi
AWS RoboMaker	`arn:aws:states:::aws-sdk:robomaker: [apiAction]`	30 September 2021	RoboMaker
AWS IAM Identity Center	`arn:aws:states:::aws-sdk:identitystore: [apiAction]`	30 September 2021	Identitystore
IAM Identity Center OIDC	`arn:aws:states:::aws-sdk:ssooidc: [apiAction]`	30 September 2021	SsoOidc
AWS Secrets Manager	`arn:aws:states:::aws-sdk:secretsmanager: [apiAction]`	30 September 2021	SecretsManager
AWS Security Token Service	`arn:aws:states:::aws-sdk:sts: [apiAction]` *** —	30 September 2021	Sts
AWS Security Hub	`arn:aws:states:::aws-sdk:securityhub: [apiAction]`	30 September 2021	SecurityHub
AWS Server Migration Service	`arn:aws:states:::aws-sdk:sms: [apiAction]`	30 September 2021	Sms

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
AWS Service Catalog	`arn:aws:states:::aws-sdk:servicecatalog: [apiAction]`	30 September 2021	ServiceCatalog
AWS Service Catalog AppRegistry	`arn:aws:states:::aws-sdk:servicecatalogappregistry: [apiAction]`	30 September 2021	ServiceCatalogAppRegistry
AWS Shield	`arn:aws:states:::aws-sdk:shield: [apiAction]` ***	30 September 2021	Shield
AWS Signer	`arn:aws:states:::aws-sdk:signer: [apiAction]`	30 September 2021	Signer
IAM Identity Center	`arn:aws:states:::aws-sdk:sso: [apiAction]`	30 September 2021	Sso
IAM Identity Center Admin	`arn:aws:states:::aws-sdk:ssoadmin: [apiAction]`	30 September 2021	SsoAdmin
AWS Step Functions	`arn:aws:states:::aws-sdk:sfn: [apiAction]`	30 September 2021	Sfn
AWS Storage Gateway	`arn:aws:states:::aws-sdk:storagegateway: [apiAction]`	30 September 2021	StorageGateway

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
AWS Support	`arn:aws:states:::aws-sdk:support: [apiAction]`	30 September 2021	Support
AWS Systems Manager Incident Manager	`arn:aws:states:::aws-sdk:ssmincidents: [apiAction]`		SsmIncidents
AWS Transfer Family	`arn:aws:states:::aws-sdk:transfer: [apiAction]`	30 September 2021	Transfer
AWS WAF	`arn:aws:states:::aws-sdk:waf: [apiAction]`	30 September 2021	Waf
AWS WAF Regional	`arn:aws:states:::aws-sdk:wafregional: [apiAction]`	30 September 2021	WafRegional
AWS WAFV2	`arn:aws:states:::aws-sdk:wafv2: [apiAction]`	30 September 2021	Wafv2
AWS Well-Architected Tool	`arn:aws:states:::aws-sdk:wellarchitected: [apiAction]`	30 September 2021	WellArchitected
AWS X-Ray	`arn:aws:states:::aws-sdk:xray: [apiAction]`	30 September 2021	XRay

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
AWS Marketplace Metering Service	`arn:aws:states:::aws-sdk:marketplacemetering: [apiAction]`	30 September 2021	MarketplaceMetering
AWS Serverless Application Repository	`arn:aws:states:::aws-sdk:serverlessapplicationrepository: [apiAction]`	30 September 2021	ServerlessApplicationRepository
AWS Identity and Access Management Access Analyzer	`arn:aws:states:::aws-sdk:accessanalyzer: [apiAction]`	30 September 2021	AccessAnalyzer
Alexa for Business	`arn:aws:states:::aws-sdk:alexaforbusiness: [apiAction]`	30 September 2021	AlexaForBusiness
Amazon API Gateway	`arn:aws:states:::aws-sdk:apigateway: [apiAction]`	30 September 2021	ApiGateway
Amazon API Gateway	`arn:aws:states:::aws-sdk:apigatewayv2: [apiAction]`	30 September 2021	ApiGatewayV2
Amazon AppIntegrations	`arn:aws:states:::aws-sdk:appintegrations: [apiAction]`	30 September 2021	AppIntegrations
Amazon AppStream 2.0	`arn:aws:states:::aws-sdk:appstream: [apiAction]`	30 September 2021	AppStream

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
Amazon AppFlow	`arn:aws:states:::aws-sdk:appflow: [apiAction]`	30 September 2021	Appflow
Amazon Athena	`arn:aws:states:::aws-sdk:athena: [apiAction]`	30 September 2021	Athena
Amazon Augmented AI	`arn:aws:states:::aws-sdk:sagemakera2iruntime: [apiAction]`	30 September 2021	SageMaker A2IRuntime
Amazon Braket	`arn:aws:states:::aws-sdk:braket: [apiAction]`	30 September 2021	Braket
Amazon Chime	`arn:aws:states:::aws-sdk:chime: [apiAction]`	30 September 2021	Chime
Amazon Chime Meetings	`arn:aws:states:::aws-sdk:chimesdkmeetings: [apiAction]`	19 April 2022	ChimeSdkMeetings
Amazon Cloud Directory	`arn:aws:states:::aws-sdk:clouddirectory: [apiAction]`	30 September 2021	CloudDirectory
Amazon CloudFront	`arn:aws:states:::aws-sdk:cloudfront: [apiAction]`	30 September 2021	CloudFront

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
Amazon CloudSearch	`arn:aws:states:::aws-sdk:cloudsearch: [apiAction]`	30 September 2021	CloudSearch
Amazon CloudWatch	`arn:aws:states:::aws-sdk:cloudwatch: [apiAction]`	30 September 2021	CloudWatch
Amazon CloudWatch Application Insights	`arn:aws:states:::aws-sdk:applicationinsights: [apiAction]`	30 September 2021	ApplicationInsights
CloudWatch Evidently	`arn:aws:states:::aws-sdk:evidently: [apiAction]`	19 April 2022	Evidently
Amazon CloudWatch Logs	`arn:aws:states:::aws-sdk:cloudwatchlogs: [apiAction]`	30 September 2021	CloudWatchLogs
Amazon CloudWatch RUM	`arn:aws:states:::aws-sdk:rum: [apiAction]`	19 April 2022	Rum
Amazon CloudWatch Synthetics	`arn:aws:states:::aws-sdk:synthetics: [apiAction]`	30 September 2021	Synthetics
Amazon CodeGuru Profiler	`arn:aws:states:::aws-sdk:codeguruprofiler: [apiAction]`	30 September 2021	CodeGuruProfiler

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
Amazon CodeGuru Reviewer	`arn:aws:states:::aws-sdk:codegurureviewer: [apiAction]`	30 September 2021	CodeGuruReviewer
Amazon Cognito	`arn:aws:states:::aws-sdk:cognitoidentity: [apiAction]`	30 September 2021	CognitoIdentity
Amazon Cognito Identity Provider	`arn:aws:states:::aws-sdk:cognitoidentityprovider: [apiAction]`	30 September 2021	CognitoIdentityProvider
Amazon Cognito Sync	`arn:aws:states:::aws-sdk:cognitosync: [apiAction]`	30 September 2021	CognitoSync
Amazon Comprehend	`arn:aws:states:::aws-sdk:comprehend: [apiAction]`	30 September 2021	Comprehend
Amazon Comprehend Medical	`arn:aws:states:::aws-sdk:comprehendmedical: [apiAction]` *** —	30 September 2021	ComprehendMedical
Amazon Connect Contact Lens	`arn:aws:states:::aws-sdk:connectcontactlens: [apiAction]`	30 September 2021	ConnectContactLens
Amazon Connect Participant Service	`arn:aws:states:::aws-sdk:connectparticipant: [apiAction]`	30 September 2021	ConnectParticipant

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
Amazon Connect	`arn:aws:states:::aws-sdk:connect: [apiAction]`	30 September 2021	Connect
Amazon Connect Voice ID	`arn:aws:states:::aws-sdk:voiceid: [apiAction]`	19 April 2022	VoiceId
Amazon Connect Wisdom	`arn:aws:states:::aws-sdk:wisdom: [apiAction]`	19 April 2022	Wisdom
Amazon Data Lifecycle Manager	`arn:aws:states:::aws-sdk:dlm: [apiAction]`	30 September 2021	Dlm
Amazon Detective	`arn:aws:states:::aws-sdk:detective: [apiAction]`	30 September 2021	Detective
Amazon DevOps Guru	`arn:aws:states:::aws-sdk:devopsguru: [apiAction]`	30 September 2021	DevOpsGuru
Amazon DocumentDB (with MongoDB compatibility)	`arn:aws:states:::aws-sdk:docdb: [apiAction]`	30 September 2021	DocDb
Amazon DynamoDB	`arn:aws:states:::aws-sdk:dynamodb: [apiAction]`	30 September 2021	DynamoDb

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
Amazon DynamoDB Streams	`arn:aws:states:::aws-sdk:dynamodbstreams: [apiAction]`	30 September 2021	DynamoDbStreams
Amazon EC2 Container Registry	`arn:aws:states:::aws-sdk:ecr: [apiAction]`	30 September 2021	Ecr
Amazon EC2 Container Service	`arn:aws:states:::aws-sdk:ecs: [apiAction]`	30 September 2021	Ecs
Amazon EC2 Systems Manager	`arn:aws:states:::aws-sdk:ssm: [apiAction]`	30 September 2021	Ssm
Amazon EMR	`arn:aws:states:::aws-sdk:emrcontainers: [apiAction]` ***	30 September 2021	EmrContainers
Amazon ElastiCache	`arn:aws:states:::aws-sdk:elasticache: [apiAction]`	30 September 2021	ElastiCache
Amazon Elastic Inference	`arn:aws:states:::aws-sdk:elasticinference: [apiAction]`	30 September 2021	ElasticInference
Amazon Elastic Block Store	`arn:aws:states:::aws-sdk:ebs: [apiAction]`	30 September 2021	Ebs
Amazon Elastic Compute Cloud	`arn:aws:states:::aws-sdk:ec2: [apiAction]`	30 September 2021	Ec2

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
Amazon Elastic Container Registry Public	`arn:aws:states:::aws-sdk:ecrpublic:` *`[apiAction]`*	30 September 2021	EcrPublic
Amazon Elastic File System	`arn:aws:states:::aws-sdk:efs:` *`[apiAction]`* ***	30 September 2021	Efs
Amazon Elastic Kubernetes Service	`arn:aws:states:::aws-sdk:eks:` *`[apiAction]`*	30 September 2021	Eks
Amazon EMR	`arn:aws:states:::aws-sdk:emr:` *`[apiAction]`*	30 September 2021	Emr
Amazon Elastic Transcoder	`arn:aws:states:::aws-sdk:elastictranscoder:` *`[apiAction]`* ***	30 September 2021	ElasticTranscoder
Amazon OpenSearch Service	`arn:aws:states:::aws-sdk:elasticsearch:` *`[apiAction]`*	30 September 2021	Elasticsearch
Amazon OpenSearch Service	`arn:aws:states:::aws-sdk:opensearch:` *`[apiAction]`*	19 April 2022	OpenSearch
Amazon EventBridge	`arn:aws:states:::aws-sdk:eventbridge:` *`[apiAction]`*	30 September 2021	EventBridge
Amazon FSx	`arn:aws:states:::aws-sdk:fsx:` *`[apiAction]`*	30 September 2021	FSx

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
Amazon Forecast Query	`arn:aws:states:::aws-sdk:forecastquery: [apiAction]`	30 September 2021	Forecastquery
Amazon Forecast Service	`arn:aws:states:::aws-sdk:forecast: [apiAction]`	30 September 2021	Forecast
Amazon Fraud Detector	`arn:aws:states:::aws-sdk:frauddetector: [apiAction]`	30 September 2021	FraudDetector
Amazon GameLift	`arn:aws:states:::aws-sdk:gamelift: [apiAction]`	30 September 2021	Amazon GameLift
Amazon GameSparks	`arn:aws:states:::aws-sdk:gamesparks: [apiAction]`	27 Juli 2022	GameSparks
Amazon S3 Glacier	`arn:aws:states:::aws-sdk:glacier: [apiAction]`	30 September 2021	Glacier
Amazon GuardDuty	`arn:aws:states:::aws-sdk:guardduty: [apiAction]`	30 September 2021	GuardDuty
AWS HealthLake	`arn:aws:states:::aws-sdk:healthlake: [apiAction]`	30 September 2021	HealthLake
Amazon Honeycode	`arn:aws:states:::aws-sdk:honeycode: [apiAction]`	30 September 2021	Honeycode

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
Amazon Inspector	`arn:aws:states:::aws-sdk:inspector: [apiAction]`	30 September 2021	Inspector
Amazon Inspector V2	`arn:aws:states:::aws-sdk:inspector2: [apiAction]`	19 April 2022	Inspector2
Amazon Interactive Video Service	`arn:aws:states:::aws-sdk:ivs: [apiAction]`	30 September 2021	Ivs
Amazon Kendra	`arn:aws:states:::aws-sdk:kendra: [apiAction]`	30 September 2021	Kendra
Amazon Kinesis	`arn:aws:states:::aws-sdk:kinesis: [apiAction]` *** —	30 September 2021	Kinesis
Amazon Kinesis Analytics	`arn:aws:states:::aws-sdk:kinesisanalytics: [apiAction]`	30 September 2021	KinesisAnalytics
Amazon Kinesis Analytics V2	`arn:aws:states:::aws-sdk:kinesisanalyticsv2: [apiAction]`	30 September 2021	KinesisAnalyticsV2
Amazon Kinesis Firehose	`arn:aws:states:::aws-sdk:firehose: [apiAction]`	30 September 2021	Firehose

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
Amazon Kinesis Video Signaling Channels	`arn:aws:states:::aws-sdk:kinesisvideosignaling:` *`[apiAction]`*	30 September 2021	KinesisVideoSignaling
Amazon Kinesis Video Streams	`arn:aws:states:::aws-sdk:kinesisvideo:` *`[apiAction]`*	30 September 2021	KinesisVideo
Amazon Kinesis Video Streams Archived Media	`arn:aws:states:::aws-sdk:kinesisvideoarchivedmedia:` *`[apiAction]`*	30 September 2021	KinesisVideoArchivedMedia
Amazon Kinesis video stream	`arn:aws:states:::aws-sdk:kinesisvideomedia:` *`[apiAction]`*	30 September 2021	KinesisVideoMedia
Amazon Lex Model Building Service	`arn:aws:states:::aws-sdk:lexmodelbuilding:` *`[apiAction]`*	30 September 2021	LexModelBuilding
Amazon Lex Model Building Service V2	`arn:aws:states:::aws-sdk:lexmodelsv2:` *`[apiAction]`*	30 September 2021	LexModelsV2
Amazon Lex	`arn:aws:states:::aws-sdk:lexruntime:` *`[apiAction]`*	30 September 2021	LexRuntime
Amazon Lex Runtime V2	`arn:aws:states:::aws-sdk:lexruntimev2:` *`[apiAction]`* *** —	30 September 2021	LexRuntimeV2

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
Amazon Lightsail	`arn:aws:states:::aws-sdk:lightsail:` *`[apiAction]`*	30 September 2021	Lightsail
Amazon Location Service	`arn:aws:states:::aws-sdk:location:` *`[apiAction]`*	30 September 2021	Location
Amazon Lookout for Equipment	`arn:aws::states:::aws-sdk:lookoutequipment:` *`[apiAction]`*	30 September 2021	LookoutEquipment
Amazon Lookout for Metrics	`arn:aws:states:::aws-sdk:lookoutmetrics:` *`[apiAction]`*	30 September 2021	LookoutMetrics
Amazon Lookout for Vision	`arn:aws:states:::aws-sdk:lookoutvision:` *`[apiAction]`*	30 September 2021	LookoutVision
Amazon MQ	`arn:aws:states:::aws-sdk:mq:` *`[apiAction]`*	30 September 2021	Mq
Amazon Macie	`arn:aws:states:::aws-sdk:macie:` *`[apiAction]`*	30 September 2021	
Amazon Macie 2	`arn:aws:states:::aws-sdk:macie2:` *`[apiAction]`*	30 September 2021	Macie2
Amazon Managed Blockchain	`arn:aws:states:::aws-sdk:managedblockchain:` *`[apiAction]`*	30 September 2021	ManagedBlockchain

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
Amazon Managed Grafana	`arn:aws:states:::aws-sdk:grafana: [apiAction]`	19 April 2022	Grafana
Amazon Managed Service for Prometheus	`arn:aws:states:::aws-sdk:amp: [apiAction]`	30 September 2021	Amp
Amazon Managed Streaming for Apache Kafka	`arn:aws:states:::aws-sdk:kafka: [apiAction]`	30 September 2021	Kafka
Amazon MSK Connect	`arn:aws:states:::aws-sdk:kafkaconnect: [apiAction]`	19 April 2022	KafkaConnect
Amazon Managed Workflows for Apache Airflow	`arn:aws:states:::aws-sdk:mwaa: [apiAction]`	30 September 2021	Mwaa
Amazon Mechanical Turk	`arn:aws:states:::aws-sdk:mturk: [apiAction]`	30 September 2021	MTurk
Amazon MemoryDB for Redis	`arn:aws:states:::aws-sdk:memorydb: [apiAction]`	19 April 2022	MemoryDB
Amazon Nimble Studio	`arn:aws:states:::aws-sdk:nimble: [apiAction]`	30 September 2021	Nimble
Amazon Personalize	`arn:aws:states:::aws-sdk:personalize: [apiAction]`	30 September 2021	Personalize

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
Amazon Personalize Events	`arn:aws:states:::aws-sdk:personalizeevents: [apiAction]`	30 September 2021	PersonalizeEvents
Amazon Personalize Runtime	`arn:aws:states:::aws-sdk:personalizeruntime: [apiAction]`	30 September 2021	PersonalizeRuntime
Amazon Pinpoint	`arn:aws:states:::aws-sdk:pinpoint: [apiAction]`	30 September 2021	Pinpoint
Amazon Pinpoint Email Service	`arn:aws:states:::aws-sdk:pinpointemail: [apiAction]`	30 September 2021	PinpointEmail
Amazon Pinpoint SMS and Voice Service	`arn:aws:states:::aws-sdk:pinpointsmsvoice: [apiAction]`	30 September 2021	PinpointSmsVoice
Amazon Pinpoint SMS and Voice V2 Service	`arn:aws:states:::aws-sdk:pinpointsmsvoicev2: [apiAction]`	27 Juli 2022	PinpointSmsVoiceV2
Amazon Polly	`arn:aws:states:::aws-sdk:polly: [apiAction]`	30 September 2021	Polly
Amazon QLDB	`arn:aws:states:::aws-sdk:qldb: [apiAction]`	30 September 2021	Qldb
Amazon QLDB Session	`arn:aws:states:::aws-sdk:qldbsession: [apiAction]`	30 September 2021	QldbSession

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
Amazon QuickSight	`arn:aws:states:::aws-sdk:quicksight:` *`[apiAction]`*	30 September 2021	QuickSight
Amazon Redshift	`arn:aws:states:::aws-sdk:redshift:` *`[apiAction]`*	30 September 2021	Redshift
Amazon Redshift Data API	`arn:aws:states:::aws-sdk:redshiftdata:` *`[apiAction]`*	30 September 2021	RedshiftData
Amazon Rekognition	`arn:aws:states:::aws-sdk:rekognition:` *`[apiAction]`*	30 September 2021	Rekognition
Amazon Relational Database Service	`arn:aws:states:::aws-sdk:rds:` *`[apiAction]`*	30 September 2021	Rds
Amazon Route 53	`arn:aws:states:::aws-sdk:route53:` *`[apiAction]`*	30 September 2021	Route53
Amazon Route 53 Recovery Control Config	`arn:aws:states:::aws-sdk:route53recoverycontrolconfig:` *`[apiAction]`*	30 September 2021	Route53RecoveryControlConfig
Amazon Route 53 Domains	`arn:aws:states:::aws-sdk:route53domains:` *`[apiAction]`*	30 September 2021	Route53Domains

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
Amazon Route 53 Resolver	`arn:aws:states:::aws-sdk:route53resolver: [apiAction]`	30 September 2021	Route53Resolver
Amazon S3 on Outposts	`arn:aws:states:::aws-sdk:s3outposts: [apiAction]`	30 September 2021	S3Outposts
Amazon SageMaker Runtime Feature Store Runtime	`arn:aws:states:::aws-sdk:sagemakerfeaturestoreruntime: [apiAction]`	30 September 2021	SageMaker RuntimeFeatureStoreRuntime
Amazon SageMaker Runtime Runtime	`arn:aws:states:::aws-sdk:sagemakerruntime: [apiAction]`	30 September 2021	SageMaker RuntimeRuntime
Amazon SageMaker	`arn:aws:states:::aws-sdk:sagemaker: [apiAction]`	30 September 2021	SageMaker Runtime
Amazon SageMaker Edge Manager	`arn:aws:states:::aws-sdk:sagemakeredge: [apiAction]`	30 September 2021	SagemakerEdge
Amazon Simple Email Service	`arn:aws:states:::aws-sdk:ses: [apiAction]`	30 September 2021	Ses
Amazon Simple Email Service V2	`arn:aws:states:::aws-sdk:sesv2: [apiAction]`	30 September 2021	SesV2

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
Amazon Simple Notification Service	`arn:aws:states:::aws-sdk:sns:` *`[apiAction]`*	30 September 2021	Sns
Amazon Simple Queue Service	`arn:aws:states:::aws-sdk:sqs:` *`[apiAction]`*	30 September 2021	Sqs
Amazon Simple Storage Service	`arn:aws:states:::aws-sdk:s3:` *`[apiAction]`* ***	30 September 2021	S3
Amazon Simple Workflow Service	`arn:aws:states:::aws-sdk:swf:` *`[apiAction]`*	30 September 2021	Swf
Amazon Textract	`arn:aws:states:::aws-sdk:textract:` *`[apiAction]`*	30 September 2021	Textract
Amazon Transcribe	`arn:aws:states:::aws-sdk:transcribe:` *`[apiAction]`*	30 September 2021	Transcribe
Amazon Translate	`arn:aws:states:::aws-sdk:translate:` *`[apiAction]`*	30 September 2021	Translate
Amazon WorkDocs	`arn:aws:states:::aws-sdk:workdocs:` *`[apiAction]`*	30 September 2021	WorkDocs
Amazon WorkMail	`arn:aws:states:::aws-sdk:workmail:` *`[apiAction]`*	30 September 2021	WorkMail

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
Amazon WorkMail Message Flow	`arn:aws:states:::aws-sdk:workmailmessageflow:` *[apiAction]*	30 September 2021	WorkMailMessageFlow
Amazon WorkSpaces	`arn:aws:states:::aws-sdk:workspaces:` *[apiAction]*	30 September 2021	WorkSpaces
Amazon WorkSpaces Web	`arn:aws:states:::aws-sdk:workspacesweb:` *[apiAction]*	19 April 2022	WorkSpacesWeb
Amplify	`arn:aws:states:::aws-sdk:amplifybackend:` *[apiAction]*	30 September 2021	AmplifyBackend
Amplify UI Builder	`arn:aws:states:::aws-sdk:amplifyuibuilder:` *[apiAction]*	19 April 2022	AmplifyUiBuilder
Application Auto Scaling	`arn:aws:states:::aws-sdk:applicationautoscaling:` *[apiAction]*	30 September 2021	ApplicationAutoScaling
Amazon EC2 Auto Scaling	`arn:aws:states:::aws-sdk:autoscaling:` *[apiAction]*	30 September 2021	Auto Scaling
CodeArtifact	`arn:aws:states:::aws-sdk:codeartifact:` *[apiAction]*	30 September 2021	Codeartifact

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
DynamoDB Accelerator	`arn:aws:states:::aws-sdk:dax:` *`[apiAction]`*	30 September 2021	Dax
EC2 Image Builder	`arn:aws:states:::aws-sdk:imagebuilder:` *`[apiAction]`*	30 September 2021	Imagebuilder
AWS Elastic Disaster Recovery	`arn:aws:states:::aws-sdk:drs:` *`[apiAction]`*	19 April 2022	Drs
Elastic Load Balancing	`arn:aws:states:::aws-sdk:elasticloadbalancing:` *`[apiAction]`*	30 September 2021	ElasticLoadBalancing
Elastic Load Balancing V2	`arn:aws:states:::aws-sdk:elasticloadbalancingv2:` *`[apiAction]`*	30 September 2021	ElasticLoadBalancingV2
MediaConnect	`arn:aws:states:::aws-sdk:mediaconnect:` *`[apiAction]`*	30 September 2021	MediaConnect
Amazon S3 Control	`arn:aws:states:::aws-sdk:s3control:` *`[apiAction]`* ***	30 September 2021	S3Control
Recycle Bin for Amazon EBS	`arn:aws:states:::aws-sdk:rbin:` *`[apiAction]`*	19 April 2022	Rbin

Nama layanan	**Task**sumber daya negara	Tanggal didukung	Awalan pengecualian
Savings Plans	`arn:aws:states:::aws-sdk:savingsplans: [apiAction]`	30 September 2021	Savingsplans
Amazon EventBridge Schema Registry	`arn:aws:states:::aws-sdk:schemas: [apiAction]`	30 September 2021	Schemas
Service Quotas	`arn:aws:states:::aws-sdk:servicequotas: [apiAction]`	30 September 2021	ServiceQuotas
AWS Snowball	`arn:aws:states:::aws-sdk:snowball: [apiAction]`	30 September 2021	Snowball

Tindakan API yang tidak didukung untuk layanan yang didukung

Tabel berikut mencantumkan tindakan API yang tidak didukung untuk integrasi layanan AWS SDK. Kolom kanan berisi tindakan API yang saat ini tidak didukung untuk layanan yang tercantum di kolom kiri.

Nama layanan	Tindakan API yang tidak didukung
AWS Application Discovery Service	• `DescribeExportConfigurations` • `ExportConfigurations`
Amazon Bedrock	• `InvokeModelWithResponseStream`
Agen untuk Amazon Bedrock Runtime	• `InvokeAgent`
AWS CodeDeploy	• `BatchGetDeploymentInstances` • `GetDeploymentInstance` • `ListDeploymentInstances`

Nama layanan	Tindakan API yang tidak didukung
	• SkipWaitTimeForInstanceTermination
Amazon Comprehend Medical	• DetectEntities
AWS Direct Connect	• AllocateConnectionOnInterconnect • DescribeConnectionLoa • DescribeConnectionsOnInterconnect • DescribeInterconnectLoa
Amazon Elastic File System	• CreateTags
Amazon Elastic Transcoder	• TestRole
Amazon EMR	• DescribeJobFlows
AWS IoT	• AttachPrincipalPolicy • ListPrincipalPolicies • DetachPrincipalPolicy • ListPolicyPrincipals • DetachPrincipalPolicy
AWS IoT Penasihat Perangkat Inti	• ListTestCases
Amazon Kinesis	• SubscribeToShard
AWS Lambda	• InvokeAsync • InvokeWithResponseStream
Amazon Lex Runtime V2	• StartConversation
AWS Elemental MediaPackage	• RotateChannelCredentials
Amazon Relational Database Service	• ExecuteSql

Nama layanan	Tindakan API yang tidak didukung
Amazon Simple Storage Service	• `SelectObjectContent`
Kontrol Amazon S3	• `SelectObjectContent`
AWS Shield	• `DeleteSubscription`
AWS Security Token Service	• `AssumeRole` • `AssumeRoleWithSAML` • `AssumeRoleWithWebIdentity`

Integrasi layanan SDK yang tidak digunakan AWS lagi

Integrasi layanan AWS SDK berikut sekarang tidak digunakan lagi:

- AWS Ponsel
- Amazon Macie

Integrasi yang dioptimalkan untuk Step Functions

Topik berikut mencakup API, parameter, dan sintaks permintaan/respons yang didukung dalam Bahasa Negara Amazon untuk mengoordinasikan layanan lain. AWS Topik juga memberikan contoh kode. Anda dapat menghubungi layanan integrasi yang dioptimalkan langsung dari Amazon States Language di `Resource` bidang `Task` negara bagian.

Anda dapat menggunakan tiga pola integrasi layanan:

- Minta Respons (default) - tunggu respons HTTP, lalu pergi ke status berikutnya
- Run a Job (`.sync`) - tunggu sampai pekerjaan selesai
- Tunggu Callback (`.waitForTaskToken`) - jeda alur kerja hingga token tugas dikembalikan

Alur Kerja Standar dan Alur Kerja Ekspres mendukung integrasi yang sama tetapi bukan pola integrasi yang sama.

- Dukungan pola integrasi yang dioptimalkan berbeda untuk setiap integrasi.

- Alur Kerja Ekspres tidak mendukung Run a Job (.sync) atau Wait for Callback (.waitForTaskToken).
- Untuk informasi selengkapnya, lihat Alur Kerja Standar vs Ekspres.

Standard Workflows

Integrasi layanan yang didukung

	Layanan	Permintaan Respon	Jalankan Job (`.sync`)	Tunggu Callback () `.waitForTaskToken`
Integrasi yang dioptimalkan	Amazon API Gateway	✓		✓
	Amazon Athena	✓	✓	
	AWS Batch	✓	✓	
	Amazon Bedrock	✓	✓	✓
	AWS CodeBuild	✓	✓	
	Amazon DynamoDB	✓		
	Amazon ECS/Fargate	✓	✓	✓
	Amazon EKS	✓	✓	✓
	Amazon EMR	✓	✓	
	Amazon EMR on EKS	✓	✓	
	Amazon EMR Serverless	✓	✓	
	Amazon EventBridge	✓		✓
	AWS Glue	✓	✓	
	AWS Glue DataBrew	✓	✓	

	Layanan	Permintaan Respon	Jalankan Job (`.sync`)	Tunggu Callback () `.waitForTaskToken`
	AWS Lambda	✓		✓
	AWS Elemental MediaConvert	✓	✓	
	Amazon SageMaker	✓	✓	
	Amazon SNS	✓		✓
	Amazon SQS	✓		✓
	AWS Step Functions	✓	✓	✓
AWS Integrasi SDK	Lebih dari dua ratus	✓		✓

Express Workflows

Integrasi layanan yang didukung

	Layanan	Permintaan Respon	Jalankan Job (`.sync`)	Tunggu Callback () `.waitForTaskToken`
Integrasi yang dioptimalkan	Amazon API Gateway	✓		
	Amazon Athena	✓		
	AWS Batch	✓		
	Amazon Bedrock	✓		

	Layanan	Permintaan Respon	Jalankan Job (`.sync`)	Tunggu Callback () `.waitForTaskToken`
	AWS CodeBuild	✓		
	Amazon DynamoDB	✓		
	Amazon ECS/Fargate	✓		
	Amazon EKS	✓		
	Amazon EMR	✓		
	Amazon EMR on EKS	✓		
	Amazon EMR Serverless	✓		
	Amazon EventBridge	✓		
	AWS Glue	✓		
	AWS Glue DataBrew	✓		
	AWS Lambda	✓		
	AWS Elemental MediaConvert	✓		
	Amazon SageMaker	✓		
	Amazon SNS	✓		
	Amazon SQS	✓		
	AWS Step Functions	✓		
AWS Integrasi SDK	Lebih dari dua ratus	✓		

Panggil API Gateway dengan Step Functions

Step Functions dapat mengontrol AWS layanan tertentu langsung dari Amazon States Language (ASL). Untuk mempelajari selengkapnya, lihat Bekerja dengan layanan yang lain dan Meneruskan parameter ke API layanan.

Bagaimana integrasi API Gateway yang Dioptimalkan berbeda dari integrasi API Gateway AWS SDK

- `apigateway:invoke:`tidak memiliki padanan dalam integrasi layanan AWS SDK. Sebagai gantinya, layanan API Gateway yang Dioptimalkan memanggil titik akhir API Gateway Anda secara langsung.

Anda dapat menggunakan Amazon API Gateway untuk membuat, menerbitkan, memelihara, dan memantau API HTTP dan REST. Untuk mengintegrasikan dengan API Gateway, Anda menentukan status `Task` di Step Functions yang langsung memanggil HTTP API Gateway atau titik akhir REST API Gateway, tanpa menulis kode atau mengandalkan infrastruktur lainnya. Ketentuan status `Task` mencakup semua informasi yang diperlukan untuk panggilan API. Anda juga dapat memilih metode otorisasi yang berbeda.

Note

Step Functions mendukung kemampuan untuk memanggil titik akhir HTTP melalui API Gateway, tetapi saat ini tidak mendukung kemampuan untuk memanggil titik akhir HTTP generik.

Dukungan fitur API Gateway

Integrasi API Gateway Step Functions mendukung beberapa, tetapi tidak semua fitur API Gateway. Untuk daftar lebih detail fitur yang didukung, lihat berikut ini.

- Didukung oleh kedua integrasi API REST API Gateway Step Functions dan API HTTP API Gateway:
  - Pengotorisasi: IAM (menggunakan Versi Tanda Tangan 4), Tidak ada Otorisasi, Pengotorisasi Lambda (berbasis parameter permintaan dan berbasis token dengan header kustom)
  - Tipe API: Wilayah

  - Manajemen API: Nama domain API Gateway API, tahap API, Jalur, Parameter Kueri, Isi Permintaan
- Didukung oleh integrasi API Step Functions API Gateway HTTP. Integrasi Step Functions API Gateway REST API yang menyediakan opsi untuk API yang dioptimalkan Edge tidak didukung.
- Tidak didukung oleh integrasi API Gateway Step Functions:
  - Pengotorisasi: Amazon Cognito, Native Open ID Connect/OAuth 2.0, header Otorisasi untuk pengotorisasi Lambda berbasis token
  - Tipe API: Privat
  - Manajemen API: Nama domain kustom

Untuk informasi selengkapnya tentang API Gateway dan API HTTP dan REST, lihat berikut ini.

- Halaman Konsep Amazon API Gateway.
- Memilih antara API HTTP dan API REST di panduan developer API Gateway.

Format Permintaan

Saat Anda membuat ketentuan status `Task`, Step Functions memvalidasi parameter, membangun URL yang diperlukan untuk melakukan panggilan, kemudian memanggil API. Respons meliputi kode status HTTP, header dan isi respons. Format permintaan memiliki parameter yang diperlukan dan opsional.

Parameter permintaan yang dibutuhkan

- `ApiEndpoint`
  - Tipe: `String`
  - Nama host dari URL API Gateway. Formatnya adalah *`<API ID>`*`.execute-api.`*`<region>`*`.amazonaws.com`.

    ID API hanya dapat berisi kombinasi karakter alfanumerik berikut ini: `0123456789abcdefghijklmnopqrstuvwxyz`
- `Method`
  - Tipe: `Enum`
  - Metode HTTP, yang harus berupa salah satu dari berikut:
    - `GET`

- POST
- PUT
- DELETE
- PATCH
- HEAD
- OPTIONS

Parameter permintaan opsional.

- Headers
  - Tipe: JSON
  - Header HTTP mengizinkan daftar nilai yang terkait dengan kunci yang sama.
- Stage
  - Tipe: String
  - Nama tahap tempat API di-deploy ke dalam API Gateway. Ini opsional untuk API HTTP yang menggunakan tahap $default.
- Path
  - Tipe: String
  - Parameter jalur yang ditambahkan setelah titik akhir API.
- QueryParameters
  - Tipe: JSON
  - String kueri hanya mengizinkan daftar nilai yang terkait dengan kunci yang sama.
- RequestBody
  - Tipe: JSON atau String
  - Isi Permintaan HTTP. Tipenya bisa berupa objek JSON atau String. RequestBody hanya didukung untuk PATCH, POST, dan metode HTTP PUT.
- AllowNullValues
  - Tipe: BOOLEAN
  - Pengaturan AllowNullValues ke true akan memungkinkan Anda untuk meneruskan nilai null seperti berikut ini:

```
{
```

```
    "NewPet": {
        "type": "turtle",
        "price": 123,
        "name": null
    }
}
```

- `AuthType`
  - Tipe: `JSON`
  - Metode autentikasi. Metode default adalah `NO_AUTH`. Nilai yang diizinkan adalah:
    - `NO_AUTH`
    - `IAM_ROLE`
    - `RESOURCE_POLICY`

    Lihat Autentikasi dan otorisasi untuk informasi lebih lanjut.

> **Note**
>
> Untuk pertimbangan keamanan, kunci header HTTP berikut ini tidak diizinkan:
>
> - Apa pun berprefiks `X-Forwarded`, `X-Amz` atau `X-Amzn`.
> - `Authorization`
> - `Connection`
> - `Content-md5`
> - `Expect`
> - `Host`
> - `Max-Forwards`
> - `Proxy-Authenticate`
> - `Server`
> - `TE`
> - `Transfer-Encoding`
> - `Trailer`
> - `Upgrade`
> - `Via`

- `Www-Authenticate`

Contoh kode berikut menunjukkan cara memanggil API Gateway menggunakan Step Functions.

```
{
    "Type": "Task",
    "Resource":"arn:aws:states:::apigateway:invoke",
    "Parameters": {
        "ApiEndpoint": "example.execute-api.us-east-1.amazonaws.com",
        "Method": "GET",
        "Headers": {
            "key": ["value1", "value2"]
        },
        "Stage": "prod",
        "Path": "bills",
        "QueryParameters": {
            "billId": ["123456"]
        },
        "RequestBody": {},
        "AuthType": "NO_AUTH"
    }
}
```

Autentikasi dan otorisasi

Anda dapat menggunakan metode autentikasi berikut:

- Tidak ada otorisasi: Panggil API secara langsung tanpa metode otorisasi.
- IAM role: Dengan metode ini, Step Functions menggunakan peran mesin status, menandatangani permintaan dengan Versi Tanda Tangan 4 (SigV4), lalu memanggil API.
- Kebijakan sumber daya: Step Functions mengautentikasi permintaan, dan kemudian memanggil API. Anda harus melampirkan kebijakan sumber daya ke API yang menentukan berikut ini:

  1. Mesin status yang akan memanggil API Gateway.

> **Important**
>
> Anda harus menentukan mesin status Anda untuk membatasi akses ke sana. Jika Anda tidak, setiap mesin status yang mengautentikasi permintaan API Gateway dengan autentikasi Kebijakan sumber daya ke API Anda akan diberikan akses.

2. Step Functions itu adalah layanan memanggil API Gateway: `"Service": "states.amazonaws.com"`.
3. Sumber daya yang ingin Anda akses, termasuk:
   - *`Wilayah`*.
   - *`account-id`* di wilayah yang ditentukan.
   - *`api-id`*.
   - *`stage-name`*.
   - *`HTTP-VERB`* (metode).
   - The *`resource-path-specifier`*.

Untuk contoh kebijakan sumber daya, lihat kebijakan IAM untuk Step Functions dan API Gateway.

Untuk informasi lebih lanjut tentang format sumber daya, lihat Format sumber daya dari izin untuk mengeksekusi API di API Gateway dalam Panduan Developer API Gateway.

> **Note**
>
> Kebijakan sumber daya hanya didukung untuk API REST.

Pola integrasi layanan

Integrasi API Gateway mendukung dua pola integrasi layanan:

- Minta Tanggapan, yang merupakan pola integrasi default. Ini memungkinkan Step Functions maju ke langkah berikutnya segera setelah menerima respons HTTP.
- Tunggu Panggilan Balik dengan Token Tugas (`.waitForTaskToken`), yang menunggu sampai token tugas dikembalikan dengan muatan. Untuk menggunakan `.waitForTaskToken` pola, tambahkan. waitForTaskToken ke akhir bidang Sumber daya definisi tugas Anda seperti yang ditunjukkan pada contoh berikut:

```
{
    "Type": "Task",
    "Resource":"arn:aws:states:::apigateway:invoke.waitForTaskToken",
    "Parameters": {
        "ApiEndpoint": "example.execute-api.us-east-1.amazonaws.com",
        "Method": "POST",
        "Headers": {
            "TaskToken.$": "States.Array($$.Task.Token)"
        },
        "Stage": "prod",
        "Path": "bills/add",
        "QueryParameters": {},
        "RequestBody": {
            "billId": "my-new-bill"
        },
        "AuthType": "IAM_ROLE"
    }
}
```

Format output

Parameter output berikut disediakan:

Nama	Tipe	Deskripsi
`ResponseBody`	`JSON` atau `String`	Isi respons dari panggilan API.
`Headers`	`JSON`	Header respons
`StatusCode`	`Integer`	Kode status HTTP dari respons.
`StatusText`	`String`	Teks status respons.

Contoh respons

```
{
    "ResponseBody": {
        "myBills": []
```

```
    },
    "Headers": {
        "key": ["value1", "value2"]
    },
    "StatusCode": 200,
    "StatusText": "OK"
}
```

Penanganan kesalahan

Ketika terjadi kesalahan, `error` dan `cause` dikembalikan sebagai berikut:

- Jika kode status HTTP tersedia, kesalahan akan dikembalikan dalam format `ApiGateway.<HTTP Status Code>`.
- Jika kode status HTTP tidak tersedia, kesalahan akan dikembalikan dalam format `ApiGateway.<Exception>`.

Dalam kedua kasus, `cause` dikembalikan sebagai string.

Contoh berikut menunjukkan respons tempat kesalahan telah terjadi:

```
{
    "error": "ApiGateway.403",
    "cause": "{\"message\":\"Missing Authentication Token\"}"
}
```

> **Note**
>
> Kode status 2XX menunjukkan keberhasilan, dan tidak ada kesalahan yang dikembalikan. Semua kode status lain atau pengecualian yang dibuang akan mengakibatkan kesalahan.

Untuk informasi selengkapnya, lihat:

- Konsep Amazon API Gateway dalam Panduan Developer API Gateway.
- Kebijakan IAM untuk Amazon API Gateway
- Contoh proyek yang menunjukkan cara Buat panggilan ke API Gateway

Konsep Amazon API Gateway dalam Panduan Developer API Gateway.

Panggil Athena dengan Step Functions

Step Functions dapat mengontrol AWS layanan tertentu langsung dari Amazon States Language (ASL). Untuk mempelajari selengkapnya, lihat Bekerja dengan layanan yang lain dan Meneruskan parameter ke API layanan.

Bagaimana integrasi Athena yang Dioptimalkan berbeda dari integrasi Athena SDK AWS

- Pola Jalankan Tugas (.sync) integrasi didukung.
- Tidak ada pengoptimalan untuk pola Minta Tanggapan integrasi.
- Pola Tunggu Panggilan Balik dengan Token Tugas integrasi tidak didukung.

Integrasi AWS Step Functions layanan dengan Amazon Athena memungkinkan Anda menggunakan Step Functions untuk memulai dan menghentikan eksekusi kueri, dan mendapatkan hasil kueri. Menggunakan Step Functions, Anda dapat menjalankan kueri ad-hoc atau data terjadwal, dan mengambil hasil yang menargetkan danau data S3 Anda. Athena bersifat nirserver, sehingga tidak ada infrastruktur yang disiapkan atau dikelola dan Anda hanya membayar untuk kueri yang Anda jalankan.

Untuk berintegrasi AWS Step Functions dengan Amazon Athena, Anda menggunakan API integrasi layanan Athena yang disediakan.

API integrasi layanan sama dengan API Athena yang sesuai. Tidak semua API mendukung semua pola integrasi, seperti yang ditunjukkan dalam tabel berikut.

API	Respons Permintaan	Jalankan Tugas (.sync)
`StartQueryExecution`	✓	✓
`StopQueryExecution`	✓	
`GetQueryExecution`	✓	
`GetQueryResults`	✓	

API Amazon Athena yang didukung:

> **Note**
>
> Ada kuota untuk input maksimum atau ukuran data hasil untuk tugas di Step Functions. Ini membatasi Anda untuk 256 KB data sebagai string UTF-8 yang dikodekan ketika Anda mengirim ke, atau menerima data dari, layanan lain. Lihat Kuota yang berkaitan dengan eksekusi mesin status.

- `StartQueryExecution`
  - Permintaan sintaks
  - Parameter yang didukung:
    - `ClientRequestToken`
    - `ExecutionParameters`
    - `QueryExecutionContext`
    - `QueryString`
    - `ResultConfiguration`
    - `WorkGroup`
  - `Response syntax`
- `StopQueryExecution`
  - Permintaan sintaks
  - Parameter yang didukung:
    - `QueryExecutionId`
- `GetQueryExecution`
  - Permintaan sintaks
  - Parameter yang didukung:
    - `QueryExecutionId`
  - `Response syntax`
- `GetQueryResults`
  - Permintaan sintaks
  - Parameter yang didukung:
    - `MaxResults`
    - `NextToken`

    - QueryExecutionId
- Response syntax

Berikut ini mencakup status tugas yang memulai kueri Athena.

```
"Start an Athena query": {
  "Type": "Task",
  "Resource": "arn:aws:states:::athena:startQueryExecution.sync",
  "Parameters": {
    "QueryString": "SELECT * FROM \"myDatabase\".\"myTable\" limit 1",
    "WorkGroup": "primary",
    "ResultConfiguration": {
       "OutputLocation": "s3://athenaQueryResult"
    }
  },
  "Next": "Get results of the query"
}
```

Untuk informasi tentang cara mengonfigurasi IAM izin saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Kelola AWS Batch dengan Step Functions

Step Functions dapat mengontrol AWS layanan tertentu langsung dari Amazon States Language (ASL). Untuk mempelajari selengkapnya, lihat Bekerja dengan layanan yang lain dan Meneruskan parameter ke API layanan.

Bagaimana AWS Batch integrasi yang Dioptimalkan berbeda dari integrasi AWS BatchAWS SDK

- Pola Jalankan Tugas (.sync) integrasi tersedia.

  Perhatikan bahwa tidak ada pengoptimalan untuk pola Minta Tanggapan atau Tunggu Panggilan Balik dengan Token Tugas integrasi.

AWS Batch API yang didukung:

- SubmitJob

- Permintaan sintaks
- Parameter yang didukung:
  - ArrayProperties
  - ContainerOverrides
  - DependsOn
  - JobDefinition
  - JobName
  - JobQueue
  - Parameters
  - RetryStrategy
  - Timeout
  - Tags
- Sintaks respons

> **Parameter Step Functions dalam dinyatakan dalam PascalCase**
>
> Bahkan jika API layanan asli ada di camelCase, misalnya `startSyncExecution` tindakan API, Anda menentukan parameter PascalCase, seperti:. `StateMachineArn`

Berikut ini mencakup `Task` negara bagian yang mengirimkan AWS Batch pekerjaan dan menunggu sampai selesai.

```
{
  "StartAt": "BATCH_JOB",
  "States": {
    "BATCH_JOB": {
      "Type": "Task",
      "Resource": "arn:aws:states:::batch:submitJob.sync",
      "Parameters": {
        "JobDefinition": "preprocessing",
        "JobName": "PreprocessingBatchJob",
        "JobQueue": "SecondaryQueue",
        "Parameters.$": "$.batchjob.parameters",
        "ContainerOverrides": {
          "ResourceRequirements": [
```

```
            {
              "Type": "VCPU",
              "Value": "4"
            }
          ]
        }
      },
      "End": true
    }
  }
}
```

Untuk informasi tentang cara mengonfigurasi IAM izin saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Panggilan Amazon Bedrock dengan Step Functions

Step Functions dapat mengontrol AWS layanan tertentu langsung dari Amazon States Language (ASL). Untuk mempelajari selengkapnya, lihat Bekerja dengan layanan yang lain dan Meneruskan parameter ke API layanan.

Topik

- Amazon BedrockAPI integrasi layanan
- Definisi status tugas untuk Amazon Bedrock integrasi

Amazon BedrockAPI integrasi layanan

Untuk mengintegrasikan AWS Step FunctionsAmazon Bedrock, Anda dapat menggunakan API berikut. API ini mirip dengan Amazon Bedrock API terkait, dengan beberapa perbedaan dalam bidang permintaan yang diteruskan.

Tabel berikut menjelaskan perbedaan antara setiap API integrasi layanan dan API yang sesuaiAmazon Bedrock.

Amazon BedrockAPI integrasi layanan dan Amazon Bedrock API terkait

Amazon BedrockAPI integrasi layanan	Amazon BedrockAPI yang sesuai	Perbedaan
InvokeModel Memanggil Amazon Bedrock model yang ditentukan untuk menjalankan inferensi menggunakan input yang Anda berikan di badan permintaan. Anda gunakan `InvokeModel` untuk menjalankan inferensi untuk model teks, model gambar, dan model penyematan.	InvokeModel	Badan permintaan API integrasi Amazon Bedrock layanan menyertakan parameter tambahan berikut. • `Body`- Menentukan data masukan dalam format yang ditentukan dalam header permintaan tipe konten. `Body`berisi parameter khusus untuk model target. Jika Anda menggunakan `InvokeModel` API, Anda harus menentukan `Body` parameter. Step Functions tidak memvalidasi masukan yang Anda berikan. `Body` Saat Anda menentuka n `Body` menggunakan integrasi yang Amazon Bedrock dioptimalkan, Anda dapat menentukan muatan hingga 256 KB. Jika muatan Anda melebihi 256 KB, kami sarankan Anda menggunak annya`Input`. • `Input`- Menentukan sumber untuk mengambil data input dari. Bidang opsional ini khusus untuk integrasi yang Amazon

Amazon BedrockAPI integrasi layanan	Amazon BedrockAPI yang sesuai	Perbedaan
		Bedrock dioptimalkan denganStep Functions. Di bidang ini, Anda dapat menentukan file`S3Uri`. Anda dapat menentukan baik `Body` dalam Parameter atau`Input`, tetapi tidak keduanya. Saat Anda menentukan `Input` tanpa menentuka n`ContentType` , jenis konten dari sumber data input menjadi nilai untuk`ContentType` . • `Output`— Menentukan tujuan di mana respon API ditulis. Bidang opsional ini khusus untuk integrasi yang Amazon Bedrock dioptimal kan denganStep Functions . Di bidang ini, Anda dapat menentukan file`S3Uri`. Jika Anda menentukan bidang ini, badan respons API akan diganti dengan referensi ke Amazon S3 lokasi output asli. Contoh berikut menunjukkan sintaks untuk InvokeModel

Amazon BedrockAPI integrasi layanan	Amazon BedrockAPI yang sesuai	Perbedaan
		API untuk Amazon Bedrock integrasi. `{ "ModelId": String, // required "Accept": String, // default: application/json "ContentType": String, // default: application/json "Input": { // not from Bedrock API "S3Uri": String }, "Output": { // not from Bedrock API "S3Uri": String } }`
CreateModelCustomizationJob Membuat pekerjaan fine-tuning untuk menyesuaikan model dasar.	CreateModelCustomizationJob	Tidak ada
CreateModelCustomizationJob.sinkronisasi Membuat pekerjaan fine-tuning untuk menyesuaikan model dasar.	CreateModelCustomizationJob	Tidak ada

Untuk informasi tentang cara mengonfigurasi IAM izin saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Definisi status tugas untuk Amazon Bedrock integrasi

Definisi status Tugas berikut menunjukkan bagaimana Anda dapat berintegrasi Amazon Bedrock dengan mesin status Anda. Contoh ini menunjukkan status Tugas yang mengekstrak hasil lengkap dari pemanggilan model yang ditentukan oleh jalur,. `result_one` Ini didasarkan pada parameter Inferensi untuk model pondasi. Contoh ini menggunakan Cohere Command large language model (LLM).

```
{
  "Type": "Task",
  "Resource": "arn:aws:states:::bedrock:invokeModel",
  "Parameters": {
    "ModelId": "cohere.command-text-v14",
    "Body": {
      "prompt.$": "$.prompt_one",
      "max_tokens": 250
    },
    "ContentType": "application/json",
    "Accept": "*/*"
  },
  "ResultPath": "$.result_one",
  "ResultSelector": {
    "result_one.$": "$.Body.generations[0].text"
  },
  "End": true
}
```

> **Tip**
>
> Untuk menerapkan contoh mesin status yang terintegrasi dengan Amazon Bedrock Anda Akun AWS, lihat. Lakukan AI prompt-chaining dengan Amazon Bedrock

Panggilan AWS CodeBuild dengan Step Functions

Step Functions dapat mengontrol AWS layanan tertentu langsung dari Amazon States Language (ASL). Untuk mempelajari selengkapnya, lihat Bekerja dengan layanan yang lain dan Meneruskan parameter ke API layanan.

Bagaimana CodeBuild integrasi yang Dioptimalkan berbeda dari integrasi CodeBuild AWS SDK

- Pola Jalankan Tugas (.sync) integrasi didukung.
- Setelah Anda memanggil `StopBuild` atau`StopBuildBatch`, kumpulan build atau build tidak segera dapat dihapus hingga beberapa pekerjaan internal selesai di dalam CodeBuild untuk menyelesaikan status build atau build. Jika Anda mencoba menggunakan `BatchDeleteBuilds` atau `DeleteBuildBatch` selama periode ini, kumpulan build atau build mungkin tidak dihapus. Integrasi layanan yang dioptimalkan untuk `BatchDeleteBuilds` dan `DeleteBuildBatch` menyertakan percobaan ulang internal untuk menyederhanakan kasus penggunaan penghapusan segera setelah berhenti.

Integrasi AWS Step Functions layanan AWS CodeBuild memungkinkan Anda menggunakan Step Functions untuk memicu, menghentikan, dan mengelola build, serta berbagi laporan build. Menggunakan Step Functions, Anda dapat merancang dan menjalankan alur integrasi berkesinambungan untuk memvalidasi perubahan perangkat lunak Anda untuk aplikasi.

Tidak semua API mendukung semua pola integrasi, seperti yang ditunjukkan dalam tabel berikut.

API	Respons Permintaan	Jalankan Tugas (.sync)
StartBuild	✓	✓
StopBuild	✓	
BatchDeleteBuilds	✓	
BatchGetReports	✓	
StartBuildBatch	✓	✓
StopBuildBatch	✓	
RetryBuildBatch	✓	✓
DeleteBuildBatch	✓	

> **Parameter Step Functions dalam dinyatakan dalam PascalCase**
>
> Bahkan jika API layanan asli ada di camelCase, misalnya `startSyncExecution` tindakan API, Anda menentukan parameter PascalCase, seperti:. `StateMachineArn`

CodeBuild API dan sintaks yang didukung:

- `StartBuild`
  - Minta sintaks
  - Parameter yang didukung:
    - `ProjectName`
    - `ArtifactsOverride`
    - `BuildspecOverride`
    - `CacheOverride`
    - `CertificateOverride`
    - `ComputeTypeOverride`
    - `EncryptionKeyOverride`
    - `EnvironmentTypeOverride`
    - `EnvironmentVariablesOverride`
    - `GitCloneDepthOverride`
    - `GitSubmodulesConfigOverride`
    - `IdempotencyToken`
    - `ImageOverride`
    - `ImagePullCredentialsTypeOverride`
    - `InsecureSslOverride`
    - `LogsConfigOverride`
    - `PrivilegedModeOverride`
    - `QueuedTimeoutInMinutesOverride`
    - `RegistryCredentialOverride`
    - `ReportBuildStatusOverride`
    - `SecondaryArtifactsOverride`

    - SecondarySourcesOverride
    - SecondarySourcesVersionOverride
    - ServiceRoleOverride
    - SourceAuthOverride
    - SourceLocationOverride
    - SourceTypeOverride
    - SourceVersion
    - TimeoutInMinutesOverride
  - Sintaks respons
- StopBuild
  - Minta sintaks
  - Parameter yang didukung:
    - Id
  - Sintaks respons
- BatchDeleteBuilds
  - Minta sintaks
  - Parameter yang didukung:
    - Ids
  - Sintaks respons
- BatchGetReports
  - Minta sintaks
  - Parameter yang didukung:
    - ReportArns
  - Sintaks respons
- StartBuildBatch
  - Minta sintaks
  - Parameter yang didukung:
    - ProjectName
    - ArtifactsOverride
    - BuildBatchConfigOverride

- BuildspecOverride
- BuildTimeoutInMinutesOverride
- CacheOverride
- CertificateOverride
- ComputeTypeOverride
- DebugSessionEnabled
- EncryptionKeyOverride
- EnvironmentTypeOverride
- EnvironmentVariablesOverride
- GitCloneDepthOverride
- GitSubmodulesConfigOverride
- IdempotencyToken
- ImageOverride
- ImagePullCredentialsTypeOverride
- InsecureSslOverride
- LogsConfigOverride
- PrivilegedModeOverride
- QueuedTimeoutInMinutesOverride
- RegistryCredentialOverride
- ReportBuildBatchStatusOverride
- SecondaryArtifactsOverride
- SecondarySourcesOverride
- SecondarySourcesVersionOverride
- ServiceRoleOverride
- SourceAuthOverride
- SourceLocationOverride
- SourceTypeOverride
- SourceVersion
- Sintaks respons

- StopBuildBatch

- Minta sintaks
- Parameter yang didukung:
  - `Id`
- Sintaks respons

- `RetryBuildBatch`
  - Minta sintaks
  - Parameter yang didukung:
    - `Id`
    - `IdempotencyToken`
    - `RetryType`
  - Sintaks respons
- `DeleteBuildBatch`
  - Minta sintaks
  - Parameter yang didukung:
    - `Id`
  - Sintaks respons

> **Note**
>
> Anda dapat menggunakan keturunan rekursif `JSONPath` (`..`) operator untuk `BatchDeleteBuilds`. Ini mengembalikan array, dan mengaktifkan Anda untuk mengubah bidang `Arn` dari `StartBuild`menjadi parameter `Ids` jamak, seperti yang ditunjukkan dalam contoh berikut.
>
> ```
> "BatchDeleteBuilds": {
>     "Type": "Task",
>     "Resource": "arn:aws:states:::codebuild:batchDeleteBuilds",
>     "Parameters": {
>         "Ids.$": "$.Build..Arn"
>     },
>     "Next": "MyNextState"
> },
> ```

Untuk informasi tentang cara mengonfigurasi IAM izin saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Panggil DynamoDB dengan Step Functions

Step Functions dapat mengontrol AWS layanan tertentu langsung dari Amazon States Language (ASL). Untuk mempelajari selengkapnya, lihat Bekerja dengan layanan yang lain dan Meneruskan parameter ke API layanan.

> **Note**
>
> Ada kuota untuk input maksimum atau ukuran data hasil untuk tugas di Step Functions. Ini membatasi Anda untuk 256 KB data sebagai string UTF-8 yang dikodekan ketika Anda mengirim ke, atau menerima data dari, layanan lain. Lihat Kuota yang berkaitan dengan eksekusi mesin status.

> **Bagaimana integrasi DynamoDB yang dioptimalkan berbeda dari integrasi DynamoDB SDK AWS**
>
> - Tidak ada optimasi untuk pola Minta Tanggapan integrasi.
> - Pola Tunggu Panggilan Balik dengan Token Tugas integrasi tidak didukung.
> - Hanya `GetItem`,, `PutItem``UpdateItem`, dan tindakan `DeleteItem`API yang tersedia melalui integrasi yang dioptimalkan. Tindakan API lainnya, seperti `CreateTable`tersedia menggunakan integrasi DynamoDB AWS SDK.

API Amazon DynamoDB dan sintaksis yang Didukung:

- `GetItem`
  - Minta sintaks
  - Parameter yang didukung:
    - `Key`
    - `TableName`
    - `AttributesToGet`
    - `ConsistentRead`

    - ExpressionAttributeNames
    - ProjectionExpression
    - ReturnConsumedCapacity
  - Sintaks respons
- PutItem
  - Minta sintaks
  - Parameter yang didukung:
    - Item
    - TableName
    - ConditionalOperator
    - ConditionExpression
    - Expected
    - ExpressionAttributeNames
    - ExpressionAttributeValues
    - ReturnConsumedCapacity
    - ReturnItemCollectionMetrics
    - ReturnValues
  - Sintaks respons
- DeleteItem
  - Minta sintaks
  - Parameter yang didukung:
    - Key
    - TableName
    - ConditionalOperator
    - ConditionExpression
    - Expected
    - ExpressionAttributeNames
    - ExpressionAttributeValues
    - ReturnConsumedCapacity
    - ReturnItemCollectionMetrics

    - ReturnValues
  - Sintaks respons
- UpdateItem
  - Minta sintaks
  - Parameter yang didukung:
    - Key
    - TableName
    - AttributeUpdates
    - ConditionalOperator
    - ConditionExpression
    - Expected
    - ExpressionAttributeNames
    - ExpressionAttributeValues
    - ReturnConsumedCapacity
    - ReturnItemCollectionMetrics
    - ReturnValues
    - UpdateExpression
  - Sintaks respons

> **Parameter Step Functions dalam dinyatakan dalam PascalCase**
>
> Bahkan jika API layanan asli ada di camelCase, misalnya `startSyncExecution` tindakan API, Anda menentukan parameter PascalCase, seperti:. `StateMachineArn`

Berikut ini adalah status `Task` yang mengambil pesan dari DynamoDB.

```
    "Read Next Message from DynamoDB": {
      "Type": "Task",
      "Resource": "arn:aws:states:::dynamodb:getItem",
      "Parameters": {
        "TableName": "TransferDataRecords-DDBTable-3I41R5L5EAGT",
        "Key": {
          "MessageId": {"S.$": "$.List[0]"}
```

```
        }
      },
      "ResultPath": "$.DynamoDB",
      "Next": "Send Message to SQS"
    },
```

Untuk melihat keadaan ini dalam contoh pekerjaan, lihat contoh proyek Mentransfer catatan data (Lambda,DynamoDB,Amazon SQS).

Untuk informasi tentang cara mengonfigurasi IAM izin saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Kelola Amazon ECS atau Tugas Fargate dengan Step Functions

Step Functions dapat mengontrol AWS layanan tertentu langsung dari Amazon States Language (ASL). Untuk mempelajari selengkapnya, lihat Bekerja dengan layanan yang lain dan Meneruskan parameter ke API layanan.

Bagaimana integrasi Amazon ECS/Fargate yang Dioptimalkan berbeda dari integrasi Amazon ECS atau Fargate SDK AWS

- Pola Jalankan Tugas (.sync) integrasi didukung.
- `ecs:runTask`dapat mengembalikan respons HTTP 200, tetapi memiliki `Failures` bidang yang tidak kosong sebagai berikut:
  - Request Response: Kembalikan respons dan jangan gagal tugas. Ini sama dengan tidak ada optimasi.
  - Jalankan Token Job atau Tugas: Jika `Failures` bidang yang tidak kosong ditemukan, tugas gagal dengan `AmazonECS.Unknown` kesalahan.

Amazon ECS/API Fargate dan sintaksis yang didukung:

Parameter Step Functions dalam dinyatakan dalam PascalCase

Bahkan jika API layanan asli ada di camelCase, misalnya `startSyncExecution` tindakan API, Anda menentukan parameter PascalCase, seperti:. `StateMachineArn`

- `RunTask` memulai tugas baru menggunakan ketentuan tugas yang ditentukan.
  - Minta sintaks
  - Parameter yang didukung:
    - `Cluster`
    - `Group`
    - `LaunchType`
    - `NetworkConfiguration`
    - `Overrides`
    - `PlacementConstraints`
    - `PlacementStrategy`
    - `PlatformVersion`
    - `PropagateTags`
    - `TaskDefinition`
    - `EnableExecuteCommand`
  - Sintaks respons

Meneruskan Data ke Tugas Amazon ECS

Step Functions dapat mengontrol AWS layanan tertentu langsung dari Amazon States Language (ASL). Untuk mempelajari selengkapnya, lihat Bekerja dengan layanan yang lain dan Meneruskan parameter ke API layanan.

Anda dapat menggunakan `overrides` untuk menimpa perintah default untuk kontainer, dan meneruskan input ke tugas-tugas Amazon ECS Anda. Lihat `ContainerOverride`. Dalam contoh, kita telah menggunakan JsonPath untuk meneruskan nilai ke `Task` dari input ke `Task` status.

Berikut ini mencakup status `Task` yang menjalankan tugas Amazon ECS dan menunggunya sampai selesai.

```
{
 "StartAt": "Run an ECS Task and wait for it to complete",
 "States": {
   "Run an ECS Task and wait for it to complete": {
     "Type": "Task",
     "Resource": "arn:aws:states:::ecs:runTask.sync",
     "Parameters": {
```

```
                "Cluster": "cluster-arn",
                "TaskDefinition": "job-id",
                "Overrides": {
                    "ContainerOverrides": [
                        {
                            "Name": "container-name",
                            "Command.$": "$.commands"
                        }
                    ]
                }
            },
     "End": true
    }
  }
}
```

Baris `"Command.$": "$.commands"` di `ContainerOverrides` meneruskan perintah dari input status ke kontainer.

Untuk contoh sebelumnya, masing-masing perintah akan diteruskan sebagai pembatalan kontainer jika input untuk eksekusi adalah sebagai berikut.

```
{
  "commands": [
    "test command 1",
    "test command 2",
    "test command 3"
  ]
}
```

Berikut ini mencakup status `Task` yang menjalankan tugas Amazon ECS, dan kemudian menunggu token tugas untuk dikembalikan. Lihat Tunggu Panggilan Balik dengan Token Tugas.

```
{
   "StartAt":"Manage ECS task",
   "States":{
      "Manage ECS task":{
         "Type":"Task",
         "Resource":"arn:aws:states:::ecs:runTask.waitForTaskToken",
         "Parameters":{
            "LaunchType":"FARGATE",
            "Cluster":"cluster-arn",
```

```
            "TaskDefinition":"job-id",
            "Overrides":{
               "ContainerOverrides":[
                  {
                     "Name":"container-name",
                     "Environment":[
                        {
                           "Name":"TASK_TOKEN_ENV_VARIABLE",
                           "Value.$":"$$.Task.Token"
                        }
                     ]
                  }
               ]
            }
         },
         "End":true
      }
   }
}
```

Untuk informasi tentang cara mengonfigurasi IAM izin saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Panggil Amazon EKS dengan Step Functions

Step Functions dapat mengontrol AWS layanan tertentu langsung dari Amazon States Language (ASL). Untuk mempelajari selengkapnya, lihat Bekerja dengan layanan yang lain dan Meneruskan parameter ke API layanan.

Bagaimana integrasi Amazon EKS yang Dioptimalkan berbeda dari integrasi Amazon EKS AWS SDK

- Pola Jalankan Tugas (.sync) integrasi didukung.
- Tidak ada pengoptimalan untuk pola Minta Tanggapan integrasi.
- Pola Tunggu Panggilan Balik dengan Token Tugas integrasi tidak didukung.

Untuk informasi tentang cara mengonfigurasi IAM izin saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Step Functions menyediakan dua tipe API integrasi layanan untuk mengintegrasikan dengan Amazon Elastic Kubernetes Service. Satu memungkinkan Anda menggunakan API Amazon EKS untuk membuat dan mengelola klaster Amazon EKS. Yang lain memungkinkan Anda berinteraksi dengan klaster Anda menggunakan API Kubernetes dan menjalankan tugas sebagai bagian dari alur kerja aplikasi Anda. Anda dapat menggunakan integrasi API Kubernetes dengan klaster Amazon EKS yang dibuat menggunakan Step Functions, dengan klaster Amazon EKS yang dibuat oleh alat eksctl atau konsol Amazon EKS, atau metode serupa. Untuk informasi selengkapnya, lihat Membuat klaster Amazon EKS di Panduan Pengguna Amazon EKS.

> **Note**
>
> Integrasi EKS Step Functions hanya mendukung API Kubernetes dengan akses titik akhir publik. Secara default, titik akhir server API kluster EKS memiliki akses publik. Untuk informasi selengkapnya, lihat kontrol akses titik akhir klaster Amazon EKS di Panduan Pengguna Amazon EKS.

Step Functions tidak mengakhiri klaster Amazon EKS secara otomatis jika eksekusi dihentikan. Jika mesin status Anda berhenti sebelum klaster Amazon EKS Anda telah dihentikan, klaster Anda dapat terus berjalan tanpa batas, dan dapat menarik biaya tambahan. Untuk menghindari hal ini, pastikan bahwa setiap klaster Amazon EKS yang Anda buat diakhiri dengan benar. Untuk informasi selengkapnya, lihat:

- Menghapus klaster dalam Panduan Pengguna Amazon EKS.
- Jalankan Tugas (.sync) dalam Pola Integrasi Layanan.

> **Note**
>
> Ada kuota untuk input maksimum atau ukuran data hasil untuk tugas di Step Functions. Ini membatasi Anda untuk 256 KB data sebagai string UTF-8 yang dikodekan ketika Anda mengirim ke, atau menerima data dari, layanan lain. Lihat Kuota yang berkaitan dengan eksekusi mesin status.

Integrasi API Kubernetes

Step Functions mendukung API Kubernetes berikut:

RunJob

Integrasi layanan `eks:runJob` mengizinkan Anda untuk menjalankan tugas di klaster Amazon EKS Anda. Varian `eks:runJob.sync` mengizinkan Anda menunggu tugas selesai, dan, opsional mengambil log.

Server API Kubernetes Anda harus memberikan izin ke IAM role yang digunakan oleh mesin status Anda. Untuk informasi selengkapnya, lihat Izin.

Untuk pola Jalankan Tugas (`.sync`), status tugas ditentukan melalui polling. Step Functions awalnya menjajak pendapat dengan angka sekitar 1 jajak pendapat per menit. Angka ini akhirnya melambat menjadi sekitar 1 jajak pendapat setiap 5 menit. Jika Anda memerlukan polling yang lebih sering, atau memerlukan lebih banyak kontrol atas strategi polling, Anda dapat menggunakan integrasi `eks:call` untuk mengueri status tugas.

Integrasi `eks:runJob` khusus untuk Tugas Kubernetes `batch/v1`. Untuk informasi selengkapnya, lihat Tugas dalam dokumentasi Kubernetes. Jika Anda ingin mengelola sumber daya Kubernetes lainnya, termasuk sumber daya kustom, gunakan integrasi layanan `eks:call`. Anda dapat menggunakan Step Functions untuk membangun loop polling, seperti yang ditunjukkan dalam proyek sampel the section called “Polling untuk Status Pekerjaan (Lambda,) AWS Batch”.

Parameter yang didukung meliputi:

- `ClusterName`: Nama klaster Amazon EKS yang ingin Anda panggil.
  - `Type`: `String`
  - Wajib: ya
- `CertificateAuthority`: Data sertifikat yang dikodekan Base64 yang diperlukan untuk berkomunikasi dengan klaster Anda. Anda dapat memperoleh nilai ini dari konsol Amazon EKS atau dengan menggunakan Amazon EKS DescribeClusterAPI.
  - `Type`: `String`
  - Wajib: ya
- `Endpoint`: URL titik akhir untuk server API Kubernetes Anda. Anda dapat memperoleh nilai ini dari konsol Amazon EKS atau dengan menggunakan Amazon EKS DescribeClusterAPI.
  - `Type`: `String`
  - Wajib: ya
- `Namespace`: Namespace tempat menjalankan tugas. Jika tidak tersedia, namespace `default` digunakan.

  - Type: `String`
  - Wajib: tidak
- `Job`: Ketentuan Tugas Kubernetes. Lihat Tugas dalam dokumentasi Kubernetes.
  - Type: `JSON` atau `String`
  - Wajib: ya
- `LogOptions`: Satu set pilihan untuk mengontrol pengambilan opsional log. Hanya berlaku jika pola integrasi layanan Jalankan Tugas (.sync) digunakan untuk menunggu penyelesaian tugas.
  - Type: `JSON`
  - Wajib: tidak
  - Log termasuk dalam respons dalam kunci `logs`. Mungkin ada beberapa pod dalam tugas, masing-masing dengan beberapa kontainer.

```
{
  ...
  "logs": {
    "pods": {
      "pod1": {
        "containers": {
          "container1": {
            "log": <log>
          },
          ...
        }
      },
      ...
    }
  }
```

  - Pengambilan log dilakukan dengan dasar upaya-terbaik. Jika ada kesalahan mengambil log, menggantikan bidang `log` akan ada bidang `error` dan `cause`.
- `LogOptions.RetrieveLogs`: Aktifkan pengambilan log setelah tugas selesai. Secara default, log tidak diambil.
  - Type: `Boolean`
  - Wajib: tidak
- `LogOptions.RawLogs`: Jika `RawLogs` diatur ke BETUL, log akan dikembalikan sebagai string mentah tanpa mencoba untuk mengurainya ke JSON. Secara default, log dideserialisasikan ke JSON jika memungkinkan. Dalam beberapa kasus penguraian tersebut dapat memperkenalkan

perubahan yang tidak diinginkan, seperti membatasi ketepatan angka yang mengandung banyak digit.

- `Type`: `Boolean`
- Wajib: tidak

- `LogOptions.LogParameters`: API Log Baca API Kubernetes API mendukung parameter kueri untuk mengontrol pengambilan log. Misalnya, Anda dapat menggunakan `tailLines` atau `limitBytes` untuk membatasi ukuran log yang diambil dan tetap berada dalam kuota ukuran data Step Functions. Untuk informasi selengkapnya, lihat bagian Log baca dari Referensi API Kubernetes.
  - `Type:` Peta `String` ke `List of Strings`
  - Wajib: tidak
  - Contoh:

```
"LogParameters": {
  "tailLines": [ "6" ]
}
```

Contoh berikut mencakup status `Task` yang menjalankan tugas, menunggunya selesai, kemudian mengambil log tugas:

```
{
  "StartAt": "Run a job on EKS",
  "States": {
    "Run a job on EKS": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:runJob.sync",
      "Parameters": {
        "ClusterName": "MyCluster",
        "CertificateAuthority": "ANPAJ2UCCR6DPCEXAMPLE",
        "Endpoint": "https://AKIAIOSFODNN7EXAMPLE.yl4.us-east-1.eks.amazonaws.com",
        "LogOptions": {
          "RetrieveLogs": true
        },
        "Job": {
          "apiVersion": "batch/v1",
          "kind": "Job",
          "metadata": {
            "name": "example-job"
```

```
          },
          "spec": {
            "backoffLimit": 0,
            "template": {
              "metadata": {
                "name": "example-job"
              },
              "spec": {
                "containers": [
                  {
                    "name": "pi-2000",
                    "image": "perl",
                    "command": [ "perl" ],
                    "args": [
                      "-Mbignum=bpi",
                      "-wle",
                      "print bpi(2000)"
                    ]
                  }
                ],
                "restartPolicy": "Never"
              }
            }
          }
        },
        "End": true
      }
    }
}
```

`Call`

Integrasi layanan `eks:call` mengizinkan Anda untuk menggunakan API Kubernetes untuk membaca dan menulis objek sumber daya Kubernetes melalui titik akhir API Kubernetes.

Server API Kubernetes Anda harus memberikan izin ke IAM role yang digunakan oleh mesin status Anda. Untuk informasi selengkapnya, lihat Izin.

Untuk informasi selengkapnya tentang operasi yang tersedia, lihat Referensi API Kubernetes.

Parameter yang didukung untuk `Call` termasuk:

- `ClusterName`: Nama klaster Amazon EKS yang ingin Anda panggil.

  - Type: String
  - Diperlukan: Ya
- `CertificateAuthority`: Data sertifikat yang dikodekan Base64 yang diperlukan untuk berkomunikasi dengan klaster Anda. Anda dapat memperoleh nilai ini dari konsol Amazon EKS atau dengan menggunakan Amazon EKS DescribeClusterAPI.
  - Type: `String`
  - Diperlukan: Ya
- `Endpoint`: URL titik akhir untuk server API Kubernetes Anda. Anda dapat menemukan nilai ini di konsol Amazon EKS atau dengan menggunakan DescribeCluster API Amazon EKS.
  - Type: `String`
  - Diperlukan: Ya
- `Method`: Metode HTTP permintaan Anda. Salah satu: `GET`, `POST`, `PUT`, `DELETE`, `HEAD`, atau `PATCH`.
  - Type: `String`
  - Diperlukan: Ya
- `Path`: Jalur HTTP dari operasi API REST Kubernetes.
  - Type: `String`
  - Diperlukan: Ya
- `QueryParameters`: Parameter kueri HTTP dari operasi API REST Kubernetes.
  - Type: Peta `String` ke `List of Strings`
  - Diperlukan: Tidak
  - Contoh:

```
"QueryParameters": {
  "labelSelector": [ "job-name=example-job" ]
}
```

- `RequestBody`: Isi pesan HTTP operasi API REST Kubernetes.
  - Type: `JSON` atau `String`
  - Diperlukan: Tidak

Berikut ini mencakup status `Task` yang menggunakan `eks:call` untuk mencantumkan pod milik tugas `example-job`.

```
{
  "StartAt": "Call EKS",
  "States": {
    "Call EKS": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:call",
      "Parameters": {
        "ClusterName": "MyCluster",
        "CertificateAuthority": "ANPAJ2UCCR6DPCEXAMPLE",
        "Endpoint": "https://444455556666.yl4.us-east-1.eks.amazonaws.com",
        "Method": "GET",
        "Path": "/api/v1/namespaces/default/pods",
        "QueryParameters": {
          "labelSelector": [
            "job-name=example-job"
          ]
        }
      },
      "End": true
    }
  }
}
```

Berikut ini mencakup status `Task` yang menggunakan `eks:call` untuk menghapus tugas `example-job`, dan menetapkan `propagationPolicy` untuk memastikan pod tugas juga dihapus.

```
{
  "StartAt": "Call EKS",
  "States": {
    "Call EKS": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:call",
      "Parameters": {
        "ClusterName": "MyCluster",
        "CertificateAuthority": "ANPAJ2UCCR6DPCEXAMPLE",
        "Endpoint": "https://444455556666.yl4.us-east-1.eks.amazonaws.com",
        "Method": "DELETE",
        "Path": "/apis/batch/v1/namespaces/default/jobs/example-job",
        "QueryParameters": {
          "propagationPolicy": [
            "Foreground"
          ]
        }
```

```
      },
      "End": true
    }
  }
}
```

API Amazon EKS yang didukung

API Amazon EKS dan sintaksis yang didukung meliputi:

- `CreateCluster`
  - Permintaan sintaks
  - Sintaks respons

    Ketika klaster Amazon EKS dibuat menggunakan integrasi layanan `eks:createCluster`, IAM role ditambahkan ke tabel otorisasi RBAC Kubernetes sebagai administrator (dengan izin system:masters) Awalnya, hanya entitas IAM yang dapat melakukan panggilan ke server API Kubernetes. Untuk informasi selengkapnya, lihat:

    - Mengelola pengguna atau IAM role untuk klaster Anda di Panduan Pengguna Amazon EKS
    - Bagian Izin

    Amazon EKS menggunakan peran tertaut layanan yang berisi izin yang diperlukan Amazon EKS untuk memanggil layanan lain atas nama Anda. Jika peran tertaut layanan ini belum ada di akun Anda, Anda harus menambahkan izin `iam:CreateServiceLinkedRole` untuk IAM role yang digunakan oleh Step Functions. Untuk informasi lebih lanjut, lihat Menggunakan Peran Terkait Layanan dalam Panduan Pengguna Amazon EKS.

    IAM role yang digunakan oleh Step Functions harus memiliki izin `iam:PassRole` untuk meneruskan IAM peran klaster ke Amazon EKS. Untuk informasi selengkapnya, lihat IAM role klaster Amazon EKS di Panduan Pengguna Amazon EKS.
- `DeleteCluster`
  - Permintaan sintaks
  - Sintaks respons

    Anda harus menghapus profil Fargate atau grup simpul sebelum menghapus klaster.
- `CreateFargateProfile`
  - Permintaan sintaks

- Sintaks respons

  Amazon EKS menggunakan peran tertaut layanan yang berisi izin yang diperlukan Amazon EKS untuk memanggil layanan lain atas nama Anda. Jika peran tertaut layanan ini belum ada di akun Anda, Anda harus menambahkan izin `iam:CreateServiceLinkedRole` untuk IAM role yang digunakan oleh Step Functions. Untuk informasi lebih lanjut, lihat Menggunakan Peran Tertaut Layanan dalam Panduan Pengguna Amazon EKS.

  Amazon EKS di Fargate mungkin tidak tersedia di semua wilayah. Untuk informasi tentang ketersediaan wilayah, lihat bagian di Fargate di Panduan Pengguna Amazon EKS.

  IAM role yang digunakan oleh Step Functions harus memiliki izin `iam:PassRole` untuk meneruskan IAM role eksekusi pod ke Amazon EKS. Untuk informasi selengkapnya, lihat Peran eksekusi pod di Panduan Pengguna Amazon EKS.

- `DeleteFargateProfile`
  - Permintaan sintaks
  - Sintaks respons
- `CreateNodegroup`
  - Permintaan sintaks
  - Sintaks respons

  Amazon EKS menggunakan peran tertaut layanan yang berisi izin yang diperlukan untuk memanggil layanan lain atas nama Anda. Jika peran tertaut layanan ini belum ada di akun Anda, Anda harus menambahkan izin `iam:CreateServiceLinkedRole` ke IAM role yang digunakan oleh Step Functions. Untuk informasi lebih lanjut, lihat Menggunakan Peran Tertaut Layanan dalam Panduan Pengguna Amazon EKS.

  IAM role yang digunakan oleh Step Functions harus memiliki izin `iam:PassRole` untuk meneruskan IAM role simpul ke Amazon EKS. Untuk informasi lebih lanjut, lihat Menggunakan Peran Tertaut Layanan dalam Panduan Pengguna Amazon EKS.

- `DeleteNodegroup`
  - Permintaan sintaks
  - Sintaks respons

Berikut ini mencakup `Task` yang membuat klaster Amazon EKS.

```
{
  "StartAt": "CreateCluster.sync",
  "States": {
    "CreateCluster.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:createCluster.sync",
      "Parameters": {
        "Name": "MyCluster",
        "ResourcesVpcConfig": {
          "SubnetIds": [
            "subnet-053e7c47012341234",
            "subnet-027cfea4b12341234"
          ]
        },
        "RoleArn": "arn:aws:iam::123456789012:role/MyEKSClusterRole"
      },
      "End": true
    }
  }
}
```

Berikut ini mencakup status `Task` yang menghapus klaster Amazon EKS.

```
{
  "StartAt": "DeleteCluster.sync",
  "States": {
    "DeleteCluster.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:deleteCluster.sync",
      "Parameters": {
        "Name": "MyCluster"
      },
      "End": true
    }
  }
}
```

Berikut ini mencakup status `Task` yang membuat profil Fargate.

```
{
  "StartAt": "CreateFargateProfile.sync",
  "States": {
```

```
    "CreateFargateProfile.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:createFargateProfile.sync",
      "Parameters": {
        "ClusterName": "MyCluster",
        "FargateProfileName": "MyFargateProfile",
        "PodExecutionRoleArn": "arn:aws:iam::123456789012:role/
MyFargatePodExecutionRole",
        "Selectors": [{
            "Namespace": "my-namespace",
            "Labels": { "my-label": "my-value" }
          }]
      },
      "End": true
    }
  }
}
```

Berikut ini mencakup status `Task` yang menghapus profil Fargate.

```
{
  "StartAt": "DeleteFargateProfile.sync",
  "States": {
    "DeleteFargateProfile.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:deleteFargateProfile.sync",
      "Parameters": {
        "ClusterName": "MyCluster",
        "FargateProfileName": "MyFargateProfile"
      },
      "End": true
    }
  }
}
```

Berikut ini mencakup status `Task` yang membuat grup simpul.

```
{
  "StartAt": "CreateNodegroup.sync",
  "States": {
    "CreateNodegroup.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:createNodegroup.sync",
```

```
      "Parameters": {
        "ClusterName": "MyCluster",
        "NodegroupName": "MyNodegroup",
        "NodeRole": "arn:aws:iam::123456789012:role/MyNodeInstanceRole",
        "Subnets": ["subnet-09fb51df01234", "subnet-027cfea4b1234"]
      },
      "End": true
    }
  }
}
```

Berikut ini mencakup status `Task` yang menghapus grup simpul.

```
{
  "StartAt": "DeleteNodegroup.sync",
  "States": {
    "DeleteNodegroup.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:deleteNodegroup.sync",
      "Parameters": {
        "ClusterName": "MyCluster",
        "NodegroupName": "MyNodegroup"
      },
      "End": true
    }
  }
}
```

Izin

Ketika klaster Amazon EKS dibuat menggunakan integrasi layanan `eks:createCluster`, IAM role ditambahkan ke tabel otorisasi RBAC Kubernetes sebagai administrator, dengan izin `system:masters`. Awalnya, hanya entitas IAM yang dapat melakukan panggilan ke server API Kubernetes. Misalnya, Anda tidak akan dapat menggunakan kubectl untuk berinteraksi dengan server API Kubernetes Anda, kecuali jika Anda menggunakan peran yang sama dengan mesin status Step Functions Anda, atau jika Anda mengonfigurasi Kubernetes untuk memberikan izin ke entitas IAM tambahan. Untuk informasi selengkapnya, lihat Mengelola pengguna atau IAM role untuk klaster Anda di Panduan Pengguna Amazon EKS.

Anda dapat menambahkan izin untuk entitas IAM tambahan, seperti pengguna atau peran, dengan menambahkannya ke `aws-auth` `ConfigMap` dalam namespace kube-sistem. Jika Anda membuat klaster Anda dari Step Functions, gunakan integrasi layanan `eks:call`.

Berikut ini mencakup `Task` status yang membuat `aws-auth` `ConfigMap` dan memberikan `system:masters` izin kepada pengguna `arn:aws:iam::123456789012:user/my-user` dan peran IAM. `arn:aws:iam::123456789012:role/my-role`

```
{
  "StartAt": "Add authorized user",
  "States": {
    "Add authorized user": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:call",
      "Parameters": {
        "ClusterName": "MyCluster",
        "CertificateAuthority": "LS0tLS1CRUd...UtLS0tLQo=",
        "Endpoint": "https://444455556666.yl4.us-east-1.eks.amazonaws.com",
        "Method": "POST",
        "Path": "/api/v1/namespaces/kube-system/configmaps",
        "RequestBody": {
           "apiVersion": "v1",
           "kind": "ConfigMap",
           "metadata": {
              "name": "aws-auth",
              "namespace": "kube-system"
           },
           "data": {
             "mapUsers": "[{ \"userarn\": \"arn:aws:iam::123456789012:user/my-user\",
 \"username\": \"my-user\", \"groups\": [ \"system:masters\" ] } ]",
             "mapRoles": "[{ \"rolearn\": \"arn:aws:iam::123456789012:role/my-role\",
 \"username\": \"my-role\", \"groups\": [ \"system:masters\" ] } ]"
           }
        }
      },
      "End": true
    }
  }
```

> **Note**
>
> Anda mungkin melihat ARN untuk IAM role yang ditampilkan dalam format yang mencakup jalur /layanan-peran/, seperti `arn:aws:iam::123456789012:role/service-role/my-role`. Token jalur peran layananini tidak boleh disertakan ketika mencantumkan peran dalam `aws-auth`.

Ketika klaster Anda pertama kali dibuat `aws-auth ConfigMap` tidak akan ada, tetapi akan ditambahkan secara otomatis jika Anda membuat profil Fargate. Anda dapat mengambil nilai `aws-auth` saat ini, tambahkan izin tambahan, dan versi PUT baru. Biasanya lebih mudah membuat `aws-auth` sebelum profil Fargate.

Jika klaster Anda dibuat di luar Step Functions, Anda dapat mengonfigurasi kubectl untuk berkomunikasi dengan server API Kubernetes Anda. Kemudian, buat `aws-auth ConfigMap` baru menggunakan `kubectl apply -f aws-auth.yaml` atau edit salah satu yang sudah ada menggunakan `kubectl edit -n kube-system configmap/aws-auth`. Untuk informasi selengkapnya, lihat:

- Buat kubeconfig untuk Amazon EKS di Panduan Pengguna Amazon EKS.
- Mengelola pengguna atau IAM role untuk klaster Anda di Panduan Pengguna Amazon EKS.

Jika IAM role Anda tidak memiliki izin yang memadai di Kubernetes, `eks:call` atau integrasi layanan `eks:runJob` akan gagal dengan kesalahan berikut:

```
Error:
EKS.401

Cause:
{
  "ResponseBody": {
    "kind": "Status",
    "apiVersion": "v1",
    "metadata": {},
    "status": "Failure",
    "message": "Unauthorized",
    "reason": "Unauthorized",
    "code": 401
  },
```

```
  "StatusCode": 401,
  "StatusText": "Unauthorized"
}
```

Panggil Amazon EMR dengan Step Functions

Step Functions dapat mengontrol AWS layanan tertentu langsung dari Amazon States Language (ASL). Untuk mempelajari selengkapnya, lihat Bekerja dengan layanan yang lain dan Meneruskan parameter ke API layanan.

> **Bagaimana integrasi EMR Amazon yang Dioptimalkan berbeda dari integrasi Amazon EMR SDK AWS**
>
> Integrasi layanan EMR Amazon yang Dioptimalkan memiliki serangkaian API khusus yang membungkus API EMR Amazon yang mendasarinya, yang dijelaskan di bawah ini. Karena itu, ini berbeda secara signifikan dari integrasi layanan Amazon EMR AWS SDK. Selain itu, pola Jalankan Tugas (.sync) integrasi didukung.

Untuk berintegrasi AWS Step Functions dengan Amazon EMR, Anda menggunakan API integrasi layanan EMR Amazon yang disediakan. API integrasi layanan mirip dengan API Amazon EMR yang sesuai, dengan beberapa perbedaan dalam bidang yang diteruskan dan respons yang dikembalikan.

Step Functions tidak mengakhiri klaster Amazon EMR secara otomatis jika eksekusi dihentikan. Jika mesin status Anda berhenti sebelum klaster Amazon EMR Anda telah dihentikan, klaster Anda dapat terus berjalan tanpa batas, dan dapat memperoleh biaya tambahan. Untuk menghindari hal ini, pastikan bahwa setiap klaster Amazon EMR yang Anda buat dihentikan dengan benar. Untuk informasi selengkapnya, lihat:

- Penghentian Klaster Kontrol dalam Panduan Pengguna Amazon EMR.
- Bagian Jalankan Tugas (.sync) Pola Integrasi Layanan.

> **Note**
>
> Untuk `emr-5.28.0`, Anda dapat menentukan parameter `StepConcurrencyLevel` saat membuat sebuah klaster untuk mengizinkan beberapa langkah untuk menjalankan secara

paralel pada satu klaster. Anda dapat menggunakan status `Map` dan `Parallel` Step Functions untuk mengirimkan pekerjaan secara paralel ke klaster.

Ketersediaan integrasi layanan Amazon EMR tergantung pada ketersediaan API Amazon EMR. Silakan periksa dokumentasi Amazon EMR untuk keterbatasan di wilayah khusus.

> **Note**
>
> Untuk integrasi dengan Amazon EMR, Step Functions memiliki frekuensi polling pekerjaan 60 detik yang dikodekan keras selama 10 menit pertama dan 300 detik setelah itu.

Tabel berikut menjelaskan perbedaan antara setiap API integrasi layanan dan API Amazon EMR yang sesuai.

API Integrasi Layanan Amazon EMR dan API Amazon EMR yang sesuai

API Integrasi Layanan Amazon EMR	API EMR yang sesuai	Perbedaan
CreateCluster Membuat dan memulai menjalankan klaster (alur kerja). Amazon EMR tertaut langsung dengan tipe IAM role unik yang dikenal sebagai peran tertaut layanan. Agar `createCluster` dan `createCluster.sync` bekerja, Anda harus mengonfigurasi izin yang diperlukan untuk membuat `AWSServiceRoleForEMRCleanup` peran yang tertaut layanan. Untuk	runJobFlow	`createCluster` menggunakan sintaks permintaan yang sama seperti runJobFlow, kecuali untuk yang berikut: • Bidang `Instances.KeepJobFlowAliveWhenNoSteps` wajib, dan harus memiliki nilai Boolean `TRUE`. • Bidang `Steps` tidak diperbolehkan. • Bidang `Instances.InstanceFleets[index].Name` harus disediakan dan harus unik jika API konektor

API Integrasi Layanan Amazon EMR	API EMR yang sesuai	Perbedaan

informasi selengkapnya tentang hal ini, termasuk pernyataan yang dapat Anda tambahkan ke kebijakan izin IAM, lihat Menggunakan Peran Tertaut Layanan untuk Amazon EMR.		`modifyInstanceFleetByName` opsional digunakan.
		• Bidang `Instances.InstanceGroups[index].Name` harus disediakan dan harus unik jika API `modifyInstanceGroupByName` opsional digunakan.
		
		Responsnya adalah ini:
		
		`{ "ClusterId": "string" }`
		
		Amazon EMR menggunakan ini:
		
		`{ "JobFlowId": "string" }`

createCluster.sync Buat dan mulai jalankan klaster (alur kerja).	runJobFlow	Sama seperti `createCluster`, tapi tunggu klaster untuk mencapai status `WAITING`.

API Integrasi Layanan Amazon EMR	API EMR yang sesuai	Perbedaan

setClusterTerminationPerlindungan
 	
Mengunci klaster (alur kerja) sehingga instans EC2 dalam klaster tidak dapat diakhiri oleh intervensi pengguna, panggilan API, atau kesalahan aliran pekerjaan.
	setTerminationProtection	Permintaan menggunakan ini:
		
		`{ "ClusterId": "string" }`
		
		Amazon EMR menggunakan ini:
		
		`{ "JobFlowIds": ["string"] }`

TerminateCluster
 	
Menutup sebuah klaster (alur kerja).
	terminateJobFlows	Permintaan menggunakan ini:
		
		`{ "ClusterId": "string" }`
		
		Amazon EMR menggunakan ini:
		
		`{ "JobFlowIds": ["string"] }`

TerminateCluster.sync Menutup sebuah klaster (alur kerja).	terminateJobFlows	Sama seperti `terminate Cluster` , tapi tunggu klaster berakhir.

API Integrasi Layanan Amazon EMR	API EMR yang sesuai	Perbedaan

AddStep
 	
Menambahkan langkah baru untuk menjalankan klaster.
 	
Secara opsional, Anda juga dapat menentukan `ExecutionRoleArn` parameter saat menggunakan API ini.
	addJobFlowLangkah-langkahnya	Permintaan menggunakan kunci `"ClusterId"` . Amazon EMR menggunakan `"JobFlowId"` . Permintaan menggunakan satu langkah.
		
		`{ "Step": <"StepConfig object"> }`
		
		Amazon EMR menggunakan ini:
		
		`{ "Steps": [<StepConfig objects>] }`
		
		Respons adalah ini:
		
		`{ "StepId": "string" }`
		
		Amazon EMR mengembalikan ini:
		
		`{ "StepIds": [<strings >] }`

API Integrasi Layanan Amazon EMR	API EMR yang sesuai	Perbedaan
AddStep.sync Menambahkan langkah baru ke klaster yang berjalan. Secara opsional, Anda juga dapat menentukan `ExecutionRoleArn` parameter saat menggunakan API ini.	addJobFlowLangkah-langkahnya	Sama seperti `addStep`, tapi menunggu langkahnya selesai.

API Integrasi Layanan Amazon EMR	API EMR yang sesuai	Perbedaan

CancelStep
 	
Membatalkan langkah yang tertunda dalam sebuah klaster yang berjalan.
	CancelSteps	Permintaan menggunakan ini:
		
		`{ "StepId": "string" }`
		
		Amazon EMR menggunakan ini:
		
		`{ "StepIds": [<strings >] }`
		
		Respons adalah ini:
		
		`{ "CancelStepsInfo": <CancelStepsInfo object> }`
		
		Amazon EMR menggunakan ini:
		
		`{ "CancelStepsInfoLi st": [<CancelStepsInfo objects>] }`

API Integrasi Layanan Amazon EMR	API EMR yang sesuai	Perbedaan
modifyInstanceFleetByName Memodifikasi target Sesuai Permintaan dan kapasitas Spot target untuk armada instans dengan `InstanceFleetName` yang ditentukan.	modifyInstanceFleet	Permintaan adalah sama seperti untuk `modifyInstanceFleet`, kecuali yang berikut ini: • Bidang `Instance.InstanceFleetId` tidak diperbolehkan. • Saat waktu aktif `InstanceFleetId` ditentukan secara otomatis oleh integrasi layanan dengan memanggil `ListInstanceFleets` dan mengurai hasilnya.

API Integrasi Layanan Amazon EMR	API EMR yang sesuai	Perbedaan

modifyInstanceGroupByName
 	
Memodifikasi jumlah simpul dan pengaturan konfigurasi dari grup instans.
	modifyInstanceGroups	Permintaan adalah ini:
		
		`{ "ClusterId": "string", "InstanceGroup": <InstanceGroupModifyConfig object> }`
		
		Amazon EMR menggunakan daftar:
		
		`{ "ClusterId": ["string"], "InstanceGroups": [<InstanceGroupModifyConfig objects>] }`
		
		Dalam objek `InstanceGroupModifyConfig`, bidang `InstanceGroupId` tidak diizinkan.
		
		Sebuah bidang baru, `InstanceGroupName`, telah ditambahkan. Saat waktu aktif `InstanceGroupId` ditentukan secara otomatis oleh integrasi layanan dengan memanggil `ListInstanceGroups` dan mengurai hasilnya.

Berikut ini mencakup status `Task` yang membuat sebuah klaster.

```
"Create_Cluster": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:createCluster.sync",
    "Parameters": {
        "Name": "MyWorkflowCluster",
        "VisibleToAllUsers": true,
        "ReleaseLabel": "emr-5.28.0",
        "Applications": [
            {
                "Name": "Hive"
            }
        ],
        "ServiceRole": "EMR_DefaultRole",
        "JobFlowRole": "EMR_EC2_DefaultRole",
        "LogUri": "s3n://aws-logs-123456789012-us-east-1/elasticmapreduce/",
        "Instances": {
            "KeepJobFlowAliveWhenNoSteps": true,
            "InstanceFleets": [
                {
                    "InstanceFleetType": "MASTER",
                    "Name": "MASTER",
                    "TargetOnDemandCapacity": 1,
                    "InstanceTypeConfigs": [
                        {
                            "InstanceType": "m4.xlarge"
                        }
                    ]
                },
                {
                    "InstanceFleetType": "CORE",
                    "Name": "CORE",
                    "TargetOnDemandCapacity": 1,
                    "InstanceTypeConfigs": [
                        {
                            "InstanceType": "m4.xlarge"
                        }
                    ]
                }
            ]
        }
    },
    "End": true
```

```
}
```

Berikut ini mencakup status `Task` yang mengaktifkan perlindungan pengakhiran.

```
"Enable_Termination_Protection": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:setClusterTerminationProtection",
    "Parameters": {
        "ClusterId.$": "$.ClusterId",
        "TerminationProtected": true
    },
    "End": true
}
```

Berikut ini mencakup status `Task` yang mengirimkan langkah untuk sebuah klaster.

```
"Step_One": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:addStep.sync",
    "Parameters": {
        "ClusterId.$": "$.ClusterId",
        "ExecutionRoleArn": "arn:aws:iam::123456789012:role/myEMR-execution-role",
        "Step": {
            "Name": "The first step",
            "ActionOnFailure": "CONTINUE",
            "HadoopJarStep": {
                "Jar": "command-runner.jar",
                "Args": [
                    "hive-script",
                    "--run-hive-script",
                    "--args",
                    "-f",
                    "s3://<region>.elasticmapreduce.samples/cloudfront/code/
Hive_CloudFront.q",
                    "-d",
                    "INPUT=s3://<region>.elasticmapreduce.samples",
                    "-d",
                    "OUTPUT=s3://<mybucket>/MyHiveQueryResults/"
                ]
            }
        }
    },
    "End": true
```

```
}
```

Berikut ini mencakup status `Task` yang membatalkan langkah.

```
"Cancel_Step_One": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:cancelStep",
    "Parameters": {
        "ClusterId.$": "$.ClusterId",
        "StepId.$": "$.AddStepsResult.StepId"
    },
    "End": true
}
```

Berikut ini mencakup status `Task` yang mengakhiri klaster.

```
"Terminate_Cluster": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:terminateCluster.sync",
    "Parameters": {
        "ClusterId.$": "$.ClusterId"
    },
    "End": true
}
```

Berikut ini mencakup status `Task` yang menaikkan dan menurunkan penskalaan klaster untuk grup instans.

```
"ModifyInstanceGroupByName": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:modifyInstanceGroupByName",
    "Parameters": {
        "ClusterId": "j-1234567890123",
        "InstanceGroupName": "MyCoreGroup",
        "InstanceGroup": {
            "InstanceCount": 8
        }
    },
    "End": true
}
```

Berikut ini mencakup status `Task` yang menaikkan dan menurunkan penskalaan klaster untuk armada instans.

```
"ModifyInstanceFleetByName": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:modifyInstanceFleetByName",
    "Parameters": {
        "ClusterId": "j-1234567890123",
        "InstanceFleetName": "MyCoreFleet",
        "InstanceFleet": {
            "TargetOnDemandCapacity": 8,
            "TargetSpotCapacity": 0
        }
    },
    "End": true
}
```

Untuk informasi tentang cara mengonfigurasi IAM izin saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Hubungi Amazon EMR di EKS dengan AWS Step Functions

Step Functions dapat mengontrol AWS layanan tertentu langsung dari Amazon States Language (ASL). Untuk mempelajari selengkapnya, lihat Bekerja dengan layanan yang lain dan Meneruskan parameter ke API layanan.

Bagaimana EMR Amazon yang Dioptimalkan pada integrasi EKS berbeda dari EMR Amazon pada integrasi EKS SDK AWS

- Pola Jalankan Tugas (.sync) integrasi didukung.
- Tidak ada pengoptimalan untuk pola Minta Tanggapan integrasi.
- Pola Tunggu Panggilan Balik dengan Token Tugas integrasi tidak didukung.

Note

Untuk integrasi dengan Amazon EMR, Step Functions memiliki frekuensi polling pekerjaan 60 detik yang dikodekan keras selama 10 menit pertama dan 300 detik setelah itu.

Untuk berintegrasi AWS Step Functions dengan Amazon EMR di EKS, gunakan Amazon EMR pada API integrasi layanan EKS. API integrasi layanan sama dengan Amazon EMR terkait pada API EKS, tetapi tidak semua API mendukung semua pola integrasi, seperti yang ditunjukkan pada tabel berikut.

API	Respons permintaan	Jalankan tugas (.sync)
CreateVirtualCluster	✓	
DeleteVirtualCluster	✓	✓
StartJobRun	✓	✓

Amazon EMR yang didukung di API EKS:

> **Note**
>
> Ada kuota untuk input maksimum atau ukuran data hasil untuk tugas di Step Functions. Ini membatasi Anda untuk 256 KB data sebagai string UTF-8 yang dikodekan ketika Anda mengirim ke, atau menerima data dari, layanan lain. Lihat Kuota yang berkaitan dengan eksekusi mesin status.

- `CreateVirtualCluster`
  - Minta sintaks
  - Parameter yang didukung
  - Sintaks respons
- `DeleteVirtualCluster`
  - Minta sintaks
  - Parameter yang didukung
  - Sintaks respons
- `StartJobRun`
  - Minta sintaks
  - Parameter yang didukung
  - Sintaks respons

Berikut ini mencakup status `Task` yang membuat sebuah klaster virtual.

```
"Create_Virtual_Cluster": {
  "Type": "Task",
  "Resource": "arn:aws:states:::emr-containers:createVirtualCluster",
  "Parameters": {
    "Name": "MyVirtualCluster",
    "ContainerProvider": {
      "Id": "EKSClusterName",
      "Type": "EKS",
      "Info": {
        "EksInfo": {
          "Namespace": "Namespace"
        }
      }
    }
  },
  "End": true
}
```

Berikut ini mencakup status `Task` yang menyerahkan tugas ke klaster virtual dan menunggunya untuk menyelesaikan.

```
"Submit_Job": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-containers:startJobRun.sync",
    "Parameters": {
      "Name": "MyJobName",
      "VirtualClusterId.$": "$.VirtualClusterId",
      "ExecutionRoleArn": "arn:aws:iam::<accountId>:role/job-execution-role",
      "ReleaseLabel": "emr-6.2.0-latest",
      "JobDriver": {
        "SparkSubmitJobDriver": {
          "EntryPoint": "s3://<mybucket>/jobs/trip-count.py",
          "EntryPointArguments": [
            "60"
          ],
          "SparkSubmitParameters": "--conf spark.driver.cores=2 --conf
 spark.executor.instances=10 --conf spark.kubernetes.pyspark.pythonVersion=3 --conf
 spark.executor.memory=10G --conf spark.driver.memory=10G --conf spark.executor.cores=1
 --conf spark.dynamicAllocation.enabled=false"
        }
      },
```

```
      "ConfigurationOverrides": {
        "ApplicationConfiguration": [
          {
            "Classification": "spark-defaults",
            "Properties": {
              "spark.executor.instances": "2",
              "spark.executor.memory": "2G"
            }
          }
        ],
        "MonitoringConfiguration": {
          "PersistentAppUI": "ENABLED",
          "CloudWatchMonitoringConfiguration": {
            "LogGroupName": "MyLogGroupName",
            "LogStreamNamePrefix": "MyLogStreamNamePrefix"
          },
          "S3MonitoringConfiguration": {
            "LogUri": "s3://<mylogsbucket>"
          }
        }
      },
      "Tags": {
        "taskType": "jobName"
      }
    },
    "End": true
}
```

Berikut ini mencakup status `Task` yang menghapus klaster virtual dan menunggu penghapusan selesai.

```
"Delete_Virtual_Cluster": {
  "Type": "Task",
  "Resource": "arn:aws:states:::emr-containers:deleteVirtualCluster.sync",
  "Parameters": {
    "Id.$": "$.VirtualClusterId"
  },
  "End": true
}
```

Untuk informasi tentang cara mengonfigurasi IAM izin saat menggunakan Step Functions dengan AWS layanan lain, lihat[Kebijakan IAM untuk layanan terintegrasi](#).

Panggilan Amazon EMR Serverless dengan Step Functions

Step Functions dapat mengontrol AWS layanan tertentu langsung dari Amazon States Language (ASL). Untuk mempelajari selengkapnya, lihat Bekerja dengan layanan yang lain dan Meneruskan parameter ke API layanan.

Bagaimana EMR Serverless integrasi yang Dioptimalkan berbeda dari integrasi EMR Serverless AWS SDK

- Integrasi EMR Serverless layanan yang Dioptimalkan memiliki serangkaian API khusus yang membungkus EMR Serverless API yang mendasarinya. Karena penyesuaian ini, EMR Serverless integrasi yang dioptimalkan berbeda secara signifikan dari integrasi layanan EMR Serverless AWS SDK. Selain itu, EMR Serverless integrasi yang dioptimalkan mendukung pola Jalankan Tugas (.sync) integrasi.
- Pola Tunggu Panggilan Balik dengan Token Tugas integrasi tidak didukung.

Dalam topik ini:

- EMR ServerlessAPI integrasi layanan
- Kasus penggunaan integrasi EMR Tanpa Server

EMR ServerlessAPI integrasi layanan

Untuk AWS Step Functions berintegrasiEMR Serverless, Anda dapat menggunakan enam API integrasi EMR Serverless layanan berikut. API integrasi layanan ini mirip dengan EMR Serverless API terkait, dengan beberapa perbedaan di bidang yang diteruskan dan dalam respons yang dikembalikan.

Tabel berikut menjelaskan perbedaan antara setiap API integrasi layanan dan API yang sesuaiEMR Serverless.

EMR ServerlessAPI integrasi layanan dan EMR Serverless API terkait

EMR ServerlessAPI integrasi layanan	EMR ServerlessAPI yang sesuai	Perbedaan
BuatAplikasi	CreateApplication	Tidak ada

EMR ServerlessAPI integrasi layanan	EMR ServerlessAPI yang sesuai	Perbedaan
Membuat aplikasi. EMR Serverlessterkait dengan jenis peran unik yang dikenal sebagai IAM peran terkait layanan. Agar `createApplication` dan `createApplication.sync` bekerja, Anda harus mengonfigurasi izin yang diperlukan untuk membuat `AWSServiceRoleForAmazonEMRServerless` peran yang tertaut layanan. Untuk informasi selengkapnya tentang hal ini, termasuk pernyataan yang dapat Anda tambahkan ke kebijakan IAM izin, lihat Menggunakan peran terkait layanan untuk. EMR Serverless		
CreateApplication.sync Membuat aplikasi.	CreateApplication	Tidak ada perbedaan antara permintaan dan tanggapan EMR Serverless API dan API integrasi EMR Serverless layanan. Namun, createApplication.sync menunggu aplikasi mencapai status. `CREATED`

EMR ServerlessAPI integrasi layanan	EMR ServerlessAPI yang sesuai	Perbedaan
MulaiAplikasi Memulai aplikasi tertentu dan menginisialisasi kapasitas awal aplikasi jika dikonfigurasi.	StartApplication	Respons EMR Serverless API tidak berisi data apa pun, tetapi respons API integrasi EMR Serverless layanan menyertakan data berikut. `{ "ApplicationId": "string" }`
StartApplication.sync Memulai aplikasi tertentu dan menginisialisasi kapasitas awal jika dikonfigurasi.	StartApplication	Respons EMR Serverless API tidak berisi data apa pun, tetapi respons API integrasi EMR Serverless layanan menyertakan data berikut. `{ "ApplicationId": "string" }` Selain itu, startApplication.sync menunggu aplikasi mencapai status. `STARTED`

EMR ServerlessAPI integrasi layanan	EMR ServerlessAPI yang sesuai	Perbedaan
StopAplikasi Menghentikan aplikasi tertentu dan melepaskan kapasitas awal jika dikonfigurasi. Semua pekerjaan yang dijadwalkan dan berjalan harus diselesai kan atau dibatalkan sebelum menghentikan aplikasi.	StopApplication	Respons EMR Serverless API tidak berisi data apa pun, tetapi respons API integrasi EMR Serverless layanan menyertakan data berikut. `{ "ApplicationId": "string" }`
StopApplication.sync Menghentikan aplikasi tertentu dan melepaskan kapasitas awal jika dikonfigurasi. Semua pekerjaan yang dijadwalkan dan berjalan harus diselesai kan atau dibatalkan sebelum menghentikan aplikasi.	StopApplication	Respons EMR Serverless API tidak berisi data apa pun, tetapi respons API integrasi EMR Serverless layanan menyertakan data berikut. `{ "ApplicationId": "string" }` Selain itu, StopApplication.sync menunggu aplikasi mencapai status. `STOPPED`

EMR ServerlessAPI integrasi layanan	EMR ServerlessAPI yang sesuai	Perbedaan
HapusAplikasi Menghapus aplikasi. Aplikasi harus dalam `CREATED` keadaan `STOPPED` atau agar dapat dihapus.	DeleteApplication	Respons EMR Serverless API tidak berisi data apa pun, tetapi respons API integrasi EMR Serverless layanan menyertakan data berikut. `{ "ApplicationId": "string" }`
DeleteApplication.sync Menghapus aplikasi. Aplikasi harus dalam `CREATED` keadaan `STOPPED` atau agar dapat dihapus.	DeleteApplication	Respons EMR Serverless API tidak berisi data apa pun, tetapi respons API integrasi EMR Serverless layanan menyertakan data berikut. `{ "ApplicationId": "string" }` Selain itu, StopApplication.sync menunggu aplikasi mencapai status. `TERMINATED`
startJobRun Memulai menjalankan pekerjaan.	StartJobRun	Tidak ada

EMR ServerlessAPI integrasi layanan	EMR ServerlessAPI yang sesuai	Perbedaan
startJobRun.sinkronisasi Memulai menjalankan pekerjaan.	StartJobRun	Tidak ada perbedaan antara permintaan dan tanggapan EMR Serverless API dan API integrasi EMR Serverless layanan. Namun, startJobR un.sync menunggu aplikasi mencapai status. `SUCCESS`
cancelJobRun Membatalkan menjalankan pekerjaan.	CancelJobRun	Tidak ada
cancelJobRun.sinkronisasi Membatalkan menjalankan pekerjaan.	CancelJobRun	Tidak ada perbedaan antara permintaan dan tanggapan EMR Serverless API dan API integrasi EMR Serverless layanan. Namun, cancelJob Run.sync menunggu aplikasi mencapai status. `CANCELLED`

Kasus penggunaan integrasi EMR Tanpa Server

Untuk integrasi EMR Serverless layanan yang Dioptimalkan, kami menyarankan Anda membuat satu aplikasi, dan kemudian menggunakan aplikasi itu untuk menjalankan beberapa pekerjaan. Misalnya, dalam mesin negara tunggal, Anda dapat menyertakan beberapa startJobRunpermintaan, yang semuanya menggunakan aplikasi yang sama. Contoh Tugas status berikut menunjukkan kasus penggunaan untuk mengintegrasikan EMR Serverless API denganStep Functions. Untuk informasi tentang kasus penggunaan lainnyaEMR Serverless, lihat Apa itu Amazon EMR Serverless.

> **Tip**
>
> Untuk menerapkan contoh mesin status yang terintegrasi dengan EMR Serverless untuk menjalankan beberapa pekerjaan ke Anda Akun AWS, lihat. Jalankan EMR Serverless pekerjaan

- Membuat aplikasi
- Memulai aplikasi
- Hentikan aplikasi
- Menghapus sebuah aplikasi
- Memulai pekerjaan di aplikasi
- Batalkan pekerjaan di aplikasi

Untuk informasi tentang cara mengonfigurasi IAM izin saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Dalam contoh yang ditunjukkan dalam kasus penggunaan berikut, ganti teks yang *dicetak miring dengan informasi* spesifik sumber daya Anda. Misalnya, ganti *yourApplicationId*dengan ID EMR Serverless aplikasi Anda, seperti`00yv7iv71inak893`.

Membuat aplikasi

Contoh status Tugas berikut membuat aplikasi menggunakan API integrasi layanan createApplication.sync.

```
"Create_Application": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:createApplication.sync",
    "Parameters": {
        "Name": "MyApplication",
        "ReleaseLabel": "emr-6.9.0",
        "Type": "SPARK"
    },
    "End": true
}
```

Memulai aplikasi

Contoh status Tugas berikut memulai aplikasi menggunakan API integrasi layanan startApplication.sync.

```
"Start_Application": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:startApplication.sync",
    "Parameters": {
        "ApplicationId": "yourApplicationId"
    },
    "End": true
}
```

Hentikan aplikasi

Contoh status Tugas berikut menghentikan aplikasi menggunakan API integrasi layanan StopApplication.sync.

```
"Stop_Application": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:stopApplication.sync",
    "Parameters": {
        "ApplicationId": "yourApplicationId"
    },
    "End": true
}
```

Menghapus sebuah aplikasi

Contoh status Tugas berikut menghapus aplikasi menggunakan API integrasi layanan DeleteApplication.sync.

```
"Delete_Application": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:deleteApplication.sync",
    "Parameters": {
        "ApplicationId": "yourApplicationId"
    },
    "End": true
}
```

Memulai pekerjaan di aplikasi

Contoh status Tugas berikut memulai pekerjaan dalam aplikasi menggunakan API integrasi layanan startJobRun.sync.

```
"Start_Job": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:startJobRun.sync",
    "Parameters": {
        "ApplicationId": "yourApplicationId",
        "ExecutionRoleArn": "arn:aws:iam::123456789012:role/myEMRServerless-execution-
role",
        "JobDriver": {
            "SparkSubmit": {
                "EntryPoint": "s3://<mybucket>/sample.py",
                "EntryPointArguments": ["1"],
                "SparkSubmitParameters": "--conf spark.executor.cores=4 --conf
 spark.executor.memory=4g --conf spark.driver.cores=2 --conf spark.driver.memory=4g --
conf spark.executor.instances=1"
            }
        }
    },
    "End": true
}
```

Batalkan pekerjaan di aplikasi

Contoh status Tugas berikut membatalkan pekerjaan dalam aplikasi menggunakan API integrasi layanan cancelJobRun.sync.

```
"Cancel_Job": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:cancelJobRun.sync",
    "Parameters": {
        "ApplicationId.$": "$.ApplicationId",
        "JobRunId.$": "$.JobRunId"
    },
    "End": true
}
```

Panggilan EventBridge dengan Step Functions

Step Functions dapat mengontrol AWS layanan tertentu langsung dari Amazon States Language (ASL). Untuk mempelajari selengkapnya, lihat Bekerja dengan layanan yang lain dan Meneruskan parameter ke API layanan.

Bagaimana EventBridge integrasi yang Dioptimalkan berbeda dari integrasi EventBridge AWS SDK

- Eksekusi ARN dan ARN mesin negara secara otomatis ditambahkan ke bidang masing-masing. `Resources` `PutEventsRequestEntry`
- Jika respons dari `PutEvents` berisi bukan nol `FailedEntryCount` maka `Task` status gagal dengan kesalahan`EventBridge.FailedEntry`.

Untuk informasi tentang cara mengonfigurasi IAM izin saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Step Functions menyediakan API integrasi layanan untuk mengintegrasikan dengan Amazon EventBridge. Hal ini memungkinkan Anda membangun aplikasi yang didorong peristiwa dengan mengirimkan peristiwa kustom langsung dari alur kerja Step Functions.

Untuk menggunakan `PutEvents` API, Anda harus membuat EventBridge aturan di akun Anda yang cocok dengan pola spesifik dari peristiwa yang akan Anda kirim. Misalnya, Anda dapat:

- Buat fungsi Lambda di akun Anda yang menerima dan mencetak peristiwa yang cocok dengan aturan. EventBridge
- Buat EventBridge aturan di akun Anda di bus acara default yang cocok dengan pola peristiwa tertentu dan menargetkan fungsi Lambda.

Lihat informasi yang lebih lengkap di:

- Menambahkan EventBridge acara Amazon dengan PutEvents di Panduan EventBridge Pengguna.
- Tunggu Panggilan Balik dengan Token Tugas dalam Pola Integrasi Layanan.

> **Note**
>
> Ada kuota untuk input maksimum atau ukuran data hasil untuk tugas di Step Functions. Ini membatasi Anda untuk 256 KB data sebagai string UTF-8 yang dikodekan ketika Anda mengirim ke, atau menerima data dari, layanan lain. Lihat Kuota yang berkaitan dengan eksekusi mesin status.

EventBridge API yang didukung

EventBridge API dan sintaks yang didukung meliputi:

- `PutEvents`
  - Permintaan sintaks
  - Parameter yang didukung:
    - `Entries`
  - Sintaks respons

Berikut ini mencakup `Task` yang mengirimkan peristiwa kustom:

```
{
  "Type": "Task",
  "Resource": "arn:aws:states:::events:putEvents",
  "Parameters": {
    "Entries": [
      {
        "Detail": {
          "Message": "MyMessage"
        },
        "DetailType": "MyDetailType",
        "EventBusName": "MyEventBus",
        "Source": "my.source"
      }
    ]
  },
  "End": true
}
```

Penanganan kesalahan

API `PutEvents` menerima array entri sebagai input, kemudian mengembalikan array entri hasil. Selama tindakan `PutEvents` berhasil, `PutEvents` akan mengembalikan respons HTTP 200, bahkan jika satu atau lebih entri gagal. `PutEvents` mengembalikan jumlah entri gagal dalam bidang `FailedEntryCount`.

Step Functions memeriksa apakah `FailedEntryCount` lebih dari nol. Jika lebih dari nol, Step Functions menggagalkan status dengan kesalahan `EventBridge.FailedEntry`. Hal ini memungkinkan Anda menggunakan penanganan kesalahan bawaan dari Step Functions pada status tugas untuk menangkap atau mencoba lagi ketika ada entri gagal, daripada mengharuskan menggunakan status tambahan untuk menganalisis `FailedEntryCount` dari respons.

> **Note**
>
> Jika Anda telah menerapkan idempotensi dan dapat dengan aman mencoba lagi pada semua entri, Anda dapat menggunakan logika coba lagi Step Functions. Step Functions tidak menghapus entri berhasil dari array input `PutEvents` sebelum mencoba lagi. Sebaliknya, itu mencoba ulang dengan array asli entri.

Kelola AWS Glue Pekerjaan dengan Step Functions

Step Functions dapat mengontrol AWS layanan tertentu langsung dari Amazon States Language (ASL). Untuk mempelajari selengkapnya, lihat Bekerja dengan layanan yang lain dan Meneruskan parameter ke API layanan.

> **Bagaimana AWS Glue integrasi yang Dioptimalkan berbeda dari integrasi AWS GlueAWS SDK**
>
> - Pola Jalankan Tugas (.sync) integrasi tersedia.
> - `JobName`Bidang diekstraksi dari permintaan dan dimasukkan ke dalam respons, yang biasanya hanya berisi`JobRunID`.

AWS Glue API yang didukung:

- `StartJobRun`

> **Parameter Step Functions dalam dinyatakan dalam PascalCase**
>
> Bahkan jika API layanan asli ada di camelCase, misalnya `startSyncExecution` tindakan API, Anda menentukan parameter PascalCase, seperti:. `StateMachineArn`

Berikut ini termasuk `Task` keadaan yang memulai AWS Glue pekerjaan.

```
"Glue StartJobRun": {
      "Type": "Task",
      "Resource": "arn:aws:states:::glue:startJobRun.sync",
      "Parameters": {
        "JobName": "GlueJob-JTrRO5l98qMG"
      },
      "Next": "ValidateOutput"
    },
```

Untuk informasi tentang cara mengonfigurasi IAM izin saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Kelola AWS Glue DataBrew Pekerjaan dengan Step Functions

Step Functions dapat mengontrol AWS layanan tertentu langsung dari Amazon States Language (ASL). Untuk mempelajari selengkapnya, lihat Bekerja dengan layanan yang lain dan Meneruskan parameter ke API layanan.

Anda dapat menggunakan DataBrew integrasi untuk menambahkan langkah pembersihan data dan normalisasi data ke dalam alur kerja analitik dan pembelajaran mesin Anda.

DataBrew API yang didukung:

- `StartJobRun`

Berikut ini mencakup `Task` status yang memulai pekerjaan permintaan-respons DataBrew.

```
"DataBrew StartJobRun": {
            "Type": "Task",
            "Resource": "arn:aws:states:::databrew:startJobRun",
            "Parameters": {
               "Name": "sample-proj-job-1"
```

```
            },
            "Next": "NEXT_STATE"
          },
```

Berikut ini mencakup `Task` status yang memulai DataBrew pekerjaan sinkronisasi.

```
"DataBrew StartJobRun": {
           "Type": "Task",
           "Resource": "arn:aws:states:::databrew:startJobRun.sync",
           "Parameters": {
              "Name": "sample-proj-job-1"
           },
           "Next": "NEXT_STATE"
          },
```

Untuk informasi tentang cara mengonfigurasi IAM izin saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Panggil Lambda dengan Step Functions

Step Functions dapat mengontrol AWS layanan tertentu langsung dari Amazon States Language (ASL). Untuk mempelajari selengkapnya, lihat Bekerja dengan layanan yang lain dan Meneruskan parameter ke API layanan.

> Bagaimana integrasi Lambda yang Dioptimalkan berbeda dari integrasi Lambda SDK AWS
>
> - `Payload`Bidang respons diuraikan dari Json yang melarikan diri ke Json.
> - Jika respons berisi bidang `FunctionError` atau pengecualian dimunculkan dalam fungsi Lambda, tugas gagal.

Untuk informasi selengkapnya tentang mengelola input, output, dan hasil status, lihat Pengolahan Input dan Output di Step Functions.

AWS Lambda API yang didukung:

- `Invoke`
  - Permintaan Sintaks

- Parameter yang Didukung
  - `ClientContext`
  - `FunctionName`
  - `InvocationType`
  - `Qualifier`
  - `Payload`
- Sintaks respons

> **Parameter Step Functions dalam dinyatakan dalam PascalCase**
>
> Bahkan jika API layanan asli ada di camelCase, misalnya `startSyncExecution` tindakan API, Anda menentukan parameter PascalCase, seperti:. `StateMachineArn`

Berikut ini mencakup status `Task` yang memanggil fungsi Lambda.

```
{
   "StartAt":"CallLambda",
   "States":{
      "CallLambda":{
         "Type":"Task",
         "Resource":"arn:aws:states:::lambda:invoke",
         "Parameters":{
            "FunctionName":"arn:aws:lambda:us-east-1:123456789012:function:MyFunction"
         },
         "End":true
      }
   }
}
```

Berikut ini mencakup status `Task` yang mengimplementasikan pola integrasi layanan callback.

```
{
   "StartAt":"GetManualReview",
   "States":{
      "GetManualReview":{
         "Type":"Task",
         "Resource":"arn:aws:states:::lambda:invoke.waitForTaskToken",
```

```
         "Parameters":{
            "FunctionName":"arn:aws:lambda:us-east-1:123456789012:function:get-model-
review-decision",
            "Payload":{
               "model.$":"$.new_model",
               "token.$":"$$.Task.Token"
            },
            "Qualifier":"prod-v1"
         },
         "End":true
      }
   }
}
```

Saat Anda menjalankan fungsi Lambda, eksekusi akan menunggu fungsi selesai. Jika Anda menjalankan fungsi Lambda dengan tugas panggilan balik, batas waktu detak jantung tidak mulai dihitung sampai setelah fungsi Lambda selesai mengeksekusi dan mengembalikan hasilnya. Selama fungsi Lambda dijalankan, batas waktu detak jantung tidak diberlakukan.

Dimungkinkan juga untuk memanggil Lambda secara asinkron menggunakan `InvocationType` parameter, seperti yang terlihat pada contoh berikut:

> **Note**
>
> Untuk pemanggilan fungsi Lambda yang asinkron, periode batas waktu detak jantung segera dimulai.

```
{

  "Comment": "A Hello World example of the Amazon States Language using Pass states",
  "StartAt": "Hello",
  "States": {
    "Hello": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "Parameters": {
        "FunctionName": "arn:aws:lambda:us-east-1:123456789012:function:echo",
        "InvocationType": "Event"
      },
      "End": true
```

```
    }
  }
}
```

Ketika `Task` hasilnya dikembalikan, output fungsi bersarang di dalam kamus metadata. Misalnya:

```
{

   "ExecutedVersion":"$LATEST",
   "Payload":"FUNCTION OUTPUT",
   "SdkHttpMetadata":{
      "HttpHeaders":{
         "Connection":"keep-alive",
         "Content-Length":"4",
         "Content-Type":"application/json",
         "Date":"Fri, 26 Mar 2021 07:42:02 GMT",
         "X-Amz-Executed-Version":"$LATEST",
         "x-amzn-Remapped-Content-Length":"0",
         "x-amzn-RequestId":"0101aa0101-1111-111a-aa55-1010aaa1010",
         "X-Amzn-Trace-Id":"root=1-1a1a000a2a2-fe0101aa10ab;sampled=0"
      },
      "HttpStatusCode":200
   },
   "SdkResponseMetadata":{
      "RequestId":"6b3bebdb-9251-453a-ae45-512d9e2bf4d3"
   },
   "StatusCode":200
}
```

Atau, Anda dapat memanggil fungsi Lambda dengan menentukan fungsi ARN langsung di bidang “Sumber Daya”. Saat Anda menjalankan fungsi Lambda dengan cara ini, Anda tidak dapat `.waitForTaskToken` menentukan, dan hasil tugas hanya berisi output fungsi.

```
{
   "StartAt":"CallFunction",
   "States":{
      "CallFunction": {
         "Type":"Task",
         "Resource":"arn:aws:lambda:us-east-1:123456789012:function:HelloFunction",
         "End": true
      }
   }
```

```
}
```

Anda dapat memanggil versi fungsi Lambda atau alias tertentu dengan menentukan pilihan tersebut di ARN di bidang `Resource`. Lihat yang berikut dalam dokumentasi Lambda:

- AWS Lambda pembuatan versi
- AWS Lambda alias

Untuk informasi tentang cara mengonfigurasi IAM izin saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Kelola AWS Elemental MediaConvert dengan Step Functions

> **Eksperimen dengan Step Functions dan MediaConvert**
>
> Pelajari cara menggunakan integrasi yang MediaConvert dioptimalkan dalam alur kerja yang mendeteksi dan menghapus bilah warna SMTPE dengan panjang yang tidak diketahui dari awal klip video. Baca posting blog dari 12 April 2024: Alur kerja kode rendah dengan AWS Elemental MediaConvert

Step Functions dapat mengontrol AWS layanan tertentu langsung dari Amazon States Language (ASL). Untuk mempelajari selengkapnya, lihat Bekerja dengan layanan yang lain dan Meneruskan parameter ke API layanan.

> **Bagaimana integrasi yang dioptimalkan berbeda dari integrasi AWS SDK standar**
>
> - Pola Jalankan Tugas (.sync) integrasi tersedia.
> - Tidak ada pengoptimalan untuk Minta Tanggapan atau pola Tunggu Panggilan Balik dengan Token Tugas integrasi.

MediaConvert API yang didukung:

- `CreateJob`
  - Permintaan sintaks
  - Parameter yang didukung:

- `Role` (Wajib)
- `Settings` (Wajib)
- `CreateJobRequest` (Opsional)

- Sintaks respons - lihat skema CreateJobResponse

Berikut ini mencakup `Task` negara bagian yang mengirimkan MediaConvert pekerjaan dan menunggu sampai selesai.

```
{
    "StartAt": "MediaConvert_CreateJob",
    "States": {
        "MediaConvert_CreateJob": {
        "Type": "Task",
        "Resource": "arn:aws:states:::mediaconvert:createJob.sync",
        "Parameters": {
            "Role": "arn:aws:iam::111122223333:role/Admin",
            "Settings": {
            "OutputGroups": [
                {
                "Outputs": [
                    {
                    "ContainerSettings": {
                        "Container": "MP4"
                    },
                    "VideoDescription": {
                        "CodecSettings": {
                        "Codec": "H_264",
                        "H264Settings": {
                            "MaxBitrate": 1000,
                            "RateControlMode": "QVBR",
                            "SceneChangeDetect": "TRANSITION_DETECTION"
                        }
                        }
                    },
                    "AudioDescriptions": [
                        {
                        "CodecSettings": {
                            "Codec": "AAC",
                            "AacSettings": {
                            "Bitrate": 96000,
                            "CodingMode": "CODING_MODE_2_0",
```

```
                            "SampleRate": 48000
                            }
                        }
                        }
                    ]
                    }
                ],
                "OutputGroupSettings": {
                    "Type": "FILE_GROUP_SETTINGS",
                    "FileGroupSettings": {
                    "Destination": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/"
                    }
                }
                }
            ],
            "Inputs": [
                {
                "AudioSelectors": {
                    "Audio Selector 1": {
                    "DefaultSelection": "DEFAULT"
                    }
                },
                "FileInput": "s3://DOC-EXAMPLE-SOURCE-BUCKET/DOC-EXAMPLE-SOURCE_FILE"
                }
            ]
            }
        },
        "End": true
        }
    }
}
```

Untuk informasi tentang cara mengonfigurasi IAM izin saat menggunakan Step Functions dengan MediaConvert, lihatKebijakan IAM untuk AWS Elemental MediaConvert.

> **Parameter Step Functions dalam dinyatakan dalam PascalCase**
>
> Bahkan jika API layanan asli ada di camelCase, misalnya `startSyncExecution` tindakan API, Anda menentukan parameter PascalCase, seperti:. `StateMachineArn`

Kelola SageMaker dengan Step Functions

Step Functions dapat mengontrol AWS layanan tertentu langsung dari Amazon States Language (ASL). Untuk mempelajari selengkapnya, lihat Bekerja dengan layanan yang lain dan Meneruskan parameter ke API layanan.

> Bagaimana SageMaker integrasi yang Dioptimalkan berbeda dari integrasi SageMaker AWS SDK
>
> - Pola Jalankan Tugas (.sync) integrasi didukung.
> - Tidak ada pengoptimalan untuk pola Minta Tanggapan integrasi.
> - Pola Tunggu Panggilan Balik dengan Token Tugas integrasi tidak didukung.

SageMaker API dan sintaks yang didukung:

- `CreateEndpoint`
  - Permintaan sintaks
  - Parameter yang didukung:
    - `EndpointConfigName`
    - `EndpointName`
    - `Tags`
  - Sintaks respons
- `CreateEndpointConfig`
  - Permintaan sintaks
  - Parameter yang didukung:
    - `EndpointConfigName`
    - `KmsKeyId`
    - `ProductionVariants`
    - `Tags`
  - Sintaks respons
- `CreateHyperParameterTuningJob`

> **Note**
>
> Tindakan API ini mendukung pola `.sync` integrasi.

- Permintaan sintaks
- Parameter yang didukung:
  - `HyperParameterTuningJobConfig`
  - `HyperParameterTuningJobName`
  - `Tags`
  - `TrainingJobDefinition`
  - `WarmStartConfig`
- Sintaks respons

- `CreateLabelingJob`

> **Note**
>
> Tindakan API ini mendukung pola `.sync` integrasi.

- Permintaan sintaks
- Parameter yang didukung:
  - `HumanTaskConfig`
  - `InputConfig`
  - `LabelAttributeName`
  - `LabelCategoryConfigS3Uri`
  - `LabelingJobAlgorithmsConfig`
  - `LabelingJobName`
  - `OutputConfig`
  - `RoleArn`
  - `StoppingConditions`
  - `Tags`

  - Sintaks respons
- `CreateModel`
  - Permintaan sintaks
  - Parameter yang didukung:
    - `Containers`
    - `EnableNetworkIsolation`
    - `ExecutionRoleArn`
    - `ModelName`
    - `PrimaryContainer`
    - `Tags`
    - `VpcConfig`
- `CreateProcessingJob`

> **Note**
>
> Tindakan API ini mendukung pola `.sync` integrasi.

  - Permintaan sintaks
  - Parameter yang didukung:
    - `AppSpecification`
    - `Environment`
    - `ExperimentConfig`
    - `NetworkConfig`
    - `ProcessingInputs`
    - `ProcessingJobName`
    - `ProcessingOutputConfig`
    - `ProcessingResources`
    - `RoleArn`
    - `StoppingCondition`
    - `Tags`
  - Sintaks respons

- `CreateTrainingJob`

> **Note**
>
> Tindakan API ini mendukung pola `.sync` integrasi.

  - Permintaan sintaks
  - Parameter yang didukung:
    - `AlgorithmSpecification`
    - `HyperParameters`
    - `InputDataConfig`
    - `OutputDataConfig`
    - `ResourceConfig`
    - `RoleArn`
    - `StoppingCondition`
    - `Tags`
    - `TrainingJobName`
    - `VpcConfig`
  - Sintaks respons
- `CreateTransformJob`

> **Note**
>
> Tindakan API ini mendukung pola `.sync` integrasi.

> **Note**
>
> AWS Step Functions tidak akan secara otomatis membuat kebijakan untuk`CreateTransformJob`. Anda harus melampirkan kebijakan sebaris untuk peran yang dibuat. Untuk informasi selengkapnya, lihat kebijakan IAM ini: `CreateTrainingJob`.

- Permintaan sintaks
- Parameter yang didukung:
  - BatchStrategy
  - Environment
  - MaxConcurrentTransforms
  - MaxPayloadInMB
  - ModelName
  - Tags
  - TransformInput
  - TransformJobName
  - TransformOutput
  - TransformResources
- Sintaks respons
- UpdateEndpoint
  - Permintaan sintaks
  - Parameter yang didukung:
    - EndpointConfigName
    - EndpointName
  - Sintaks respons

SageMaker Contoh Transform Job

Berikut ini mencakup `Task` status yang membuat pekerjaan SageMaker transformasi Amazon, menentukan lokasi `DataSource` Amazon S3 untuk dan. `TransformOutput`

```
{
"SageMaker CreateTransformJob": {
  "Type": "Task",
  "Resource": "arn:aws:states:::sagemaker:createTransformJob.sync",
  "Parameters": {
    "ModelName": "SageMakerCreateTransformJobModel-9iFBKsYti9vr",
    "TransformInput": {
      "CompressionType": "None",
```

```
      "ContentType": "text/csv",
      "DataSource": {
        "S3DataSource": {
          "S3DataType": "S3Prefix",
          "S3Uri": "s3://my-s3bucket-example-1/TransformJobDataInput.txt"
        }
      }
    },
    "TransformOutput": {
      "S3OutputPath": "s3://my-s3bucket-example-1/TransformJobOutputPath"
    },
    "TransformResources": {
      "InstanceCount": 1,
      "InstanceType": "ml.m4.xlarge"
    },
    "TransformJobName": "sfn-binary-classification-prediction"
  },
  "Next": "ValidateOutput"
},
```

SageMaker Contoh Training Job

Berikut ini mencakup `Task` negara bagian yang menciptakan pekerjaan SageMaker pelatihan Amazon.

```
{
   "SageMaker CreateTrainingJob":{
      "Type":"Task",
      "Resource":"arn:aws:states:::sagemaker:createTrainingJob.sync",
      "Parameters":{
         "TrainingJobName":"search-model",
         "ResourceConfig":{
            "InstanceCount":4,
            "InstanceType":"ml.c4.8xlarge",
            "VolumeSizeInGB":20
         },
         "HyperParameters":{
            "mode":"batch_skipgram",
            "epochs":"5",
            "min_count":"5",
            "sampling_threshold":"0.0001",
            "learning_rate":"0.025",
            "window_size":"5",
```

```
            "vector_dim":"300",
            "negative_samples":"5",
            "batch_size":"11"
         },
         "AlgorithmSpecification":{
            "TrainingImage":"...",
            "TrainingInputMode":"File"
         },
         "OutputDataConfig":{
            "S3OutputPath":"s3://bucket-name/doc-search/model"
         },
         "StoppingCondition":{
            "MaxRuntimeInSeconds":100000
         },
         "RoleArn":"arn:aws:iam::123456789012:role/docsearch-stepfunction-iam-role",
         "InputDataConfig":[
            {
               "ChannelName":"train",
               "DataSource":{
                  "S3DataSource":{
                     "S3DataType":"S3Prefix",
                     "S3Uri":"s3://bucket-name/doc-search/interim-data/training-data/",
                     "S3DataDistributionType":"FullyReplicated"
                  }
               }
            }
         ]
      },
      "Retry":[
         {
            "ErrorEquals":[
               "SageMaker.AmazonSageMakerException"
            ],
            "IntervalSeconds":1,
            "MaxAttempts":100,
            "BackoffRate":1.1
         },
         {
            "ErrorEquals":[
               "SageMaker.ResourceLimitExceededException"
            ],
            "IntervalSeconds":60,
            "MaxAttempts":5000,
            "BackoffRate":1
```

```
         },
         {
            "ErrorEquals":[
               "States.Timeout"
            ],
            "IntervalSeconds":1,
            "MaxAttempts":5,
            "BackoffRate":1
         }
      ],
      "Catch":[
         {
            "ErrorEquals":[
               "States.ALL"
            ],
            "ResultPath":"$.cause",
            "Next":"Sagemaker Training Job Error"
         }
      ],
      "Next":"Delete Interim Data Job"
   }
}
```

SageMaker Contoh Pelabelan Job

Berikut ini mencakup `Task` status yang menciptakan pekerjaan SageMaker pelabelan Amazon.

```
{
  "StartAt": "SageMaker CreateLabelingJob",
  "TimeoutSeconds": 3600,
  "States": {
    "SageMaker CreateLabelingJob": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sagemaker:createLabelingJob.sync",
      "Parameters": {
        "HumanTaskConfig": {
          "AnnotationConsolidationConfig": {
            "AnnotationConsolidationLambdaArn": "arn:aws:lambda:us-
west-2:123456789012:function:ACS-TextMultiClass"
          },
          "NumberOfHumanWorkersPerDataObject": 1,
```

```
          "PreHumanTaskLambdaArn": "arn:aws:lambda:us-west-2:123456789012:function:PRE-
TextMultiClass",
          "TaskDescription": "Classify the following text",
          "TaskKeywords": [
            "tc",
            "Labeling"
          ],
          "TaskTimeLimitInSeconds": 300,
          "TaskTitle": "Classify short bits of text",
          "UiConfig": {
            "UiTemplateS3Uri": "s3://s3bucket-example/TextClassification.template"
          },
          "WorkteamArn": "arn:aws:sagemaker:us-west-2:123456789012:workteam/private-
crowd/ExampleTesting"
        },
        "InputConfig": {
          "DataAttributes": {
            "ContentClassifiers": [
              "FreeOfPersonallyIdentifiableInformation",
              "FreeOfAdultContent"
            ]
          },
          "DataSource": {
            "S3DataSource": {
              "ManifestS3Uri": "s3://s3bucket-example/manifest.json"
            }
          }
        },
        "LabelAttributeName": "Categories",
        "LabelCategoryConfigS3Uri": "s3://s3bucket-example/labelcategories.json",
        "LabelingJobName": "example-job-name",
        "OutputConfig": {
          "S3OutputPath": "s3://s3bucket-example/output"
        },
        "RoleArn": "arn:aws:iam::123456789012:role/service-role/AmazonSageMaker-
ExecutionRole",
        "StoppingConditions": {
          "MaxHumanLabeledObjectCount": 10000,
          "MaxPercentageOfInputDatasetLabeled": 100
        }
      },
      "Next": "ValidateOutput"
    },
    "ValidateOutput": {
```

```
      "Type": "Choice",
      "Choices": [
        {
          "Not": {
            "Variable": "$.LabelingJobArn",
            "StringEquals": ""
          },
          "Next": "Succeed"
        }
      ],
      "Default": "Fail"
    },
    "Succeed": {
      "Type": "Succeed"
    },
    "Fail": {
      "Type": "Fail",
      "Error": "InvalidOutput",
      "Cause": "Output is not what was expected. This could be due to a service outage
 or a misconfigured service integration."
    }
  }
}
```

SageMaker Contoh Processing Job

Berikut ini mencakup `Task` status yang menciptakan pekerjaan SageMaker pemrosesan Amazon.

```
{
  "StartAt": "SageMaker CreateProcessingJob Sync",
  "TimeoutSeconds": 3600,
  "States": {
    "SageMaker CreateProcessingJob Sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sagemaker:createProcessingJob.sync",
      "Parameters": {
        "AppSpecification": {
          "ImageUri": "737474898029.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-scikit-
learn:0.20.0-cpu-py3"
        },
        "ProcessingResources": {
          "ClusterConfig": {
            "InstanceCount": 1,
```

```
            "InstanceType": "ml.t3.medium",
            "VolumeSizeInGB": 10
          }
        },
        "RoleArn": "arn:aws:iam::123456789012:role/SM-003-
CreateProcessingJobAPIExecutionRole",
        "ProcessingJobName.$": "$.id"
      },
      "Next": "ValidateOutput"
    },
    "ValidateOutput": {
      "Type": "Choice",
      "Choices": [
        {
          "Not": {
            "Variable": "$.ProcessingJobArn",
            "StringEquals": ""
          },
          "Next": "Succeed"
        }
      ],
      "Default": "Fail"
    },
    "Succeed": {
      "Type": "Succeed"
    },
    "Fail": {
      "Type": "Fail",
      "Error": "InvalidConnectorOutput",
      "Cause": "Connector output is not what was expected. This could be due to a
 service outage or a misconfigured connector."
    }
  }
}
```

Untuk informasi tentang cara mengonfigurasi IAM izin saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Panggil Amazon SNS dengan Step Functions

Step Functions dapat mengontrol AWS layanan tertentu langsung dari Amazon States Language (ASL). Untuk mempelajari selengkapnya, lihat Bekerja dengan layanan yang lain dan Meneruskan parameter ke API layanan.

> **Bagaimana integrasi Amazon SNS yang Dioptimalkan berbeda dari integrasi Amazon AWS SNS SDK**
>
> Tidak ada pengoptimalan untuk pola Minta Tanggapan atau Tunggu Panggilan Balik dengan Token Tugas integrasi.

API Amazon SNS yang didukung:

> **Note**
>
> Ada kuota untuk input maksimum atau ukuran data hasil untuk tugas di Step Functions. Ini membatasi Anda untuk 256 KB data sebagai string UTF-8 yang dikodekan ketika Anda mengirim ke, atau menerima data dari, layanan lain. Lihat Kuota yang berkaitan dengan eksekusi mesin status.

- `Publish`
  - Permintaan sintaks
  - Parameter yang Didukung
    - `Message`
    - `MessageAttributes`
    - `MessageStructure`
    - `PhoneNumber`
    - `Subject`
    - `TargetArn`
    - `TopicArn`
  - Sintaks respon

> **Parameter Step Functions dalam dinyatakan dalam PascalCase**
>
> Bahkan jika API layanan asli ada di camelCase, misalnya `startSyncExecution` tindakan API, Anda menentukan parameter PascalCase, seperti:. `StateMachineArn`

Berikut ini mencakup status `Task` yang menerbitkan ke topik Amazon Simple Notification Service (Amazon SNS).

```
{
 "StartAt": "Publish to SNS",
 "States": {
   "Publish to SNS": {
     "Type": "Task",
     "Resource": "arn:aws:states:::sns:publish",
     "Parameters": {
       "TopicArn": "arn:aws:sns:us-east-1:123456789012:myTopic",
       "Message.$": "$.input.message",
       "MessageAttributes": {
         "my_attribute_no_1": {
           "DataType": "String",
           "StringValue": "value of my_attribute_no_1"
         },
         "my_attribute_no_2": {
           "DataType": "String",
           "StringValue": "value of my_attribute_no_2"
         }
       }
     },
     "End": true
    }
  }
}
```

Meneruskan nilai dinamis. Anda dapat memodifikasi contoh di atas untuk secara dinamis melewati atribut dari muatan JSON ini:

```
{
"input": {
  "message": "Hello world"
},
"SNSDetails": {
  "attribute1": "some value",
  "attribute2": "some other value",
}
}
```

Tambahkan `.$` ke bidang `StringValue`:

```
"MessageAttributes": {
  "my_attribute_no_1": {
      "DataType": "String",
      "StringValue.$": "$.SNSDetails.attribute1"
  },
  "my_attribute_no_2": {
      "DataType": "String",
      "StringValue.$": "$.SNSDetails.attribute2"
  }
```

Berikut ini mencakup status `Task` yang menerbitkan topik Amazon SNS, dan kemudian menunggu token tugas dikembalikan. Lihat Tunggu Panggilan Balik dengan Token Tugas.

```
{
   "StartAt":"Send message to SNS",
   "States":{
      "Send message to SNS":{
         "Type":"Task",
         "Resource":"arn:aws:states:::sns:publish.waitForTaskToken",
         "Parameters":{
            "TopicArn":"arn:aws:sns:us-east-1:123456789012:myTopic",
            "Message":{
               "Input.$":"$",
               "TaskToken.$":"$$.Task.Token"
            }
         },
         "End":true
      }
   }
}
```

Untuk informasi tentang cara mengonfigurasi IAM izin saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Panggil Amazon SQS dengan Step Functions

Step Functions dapat mengontrol AWS layanan tertentu langsung dari Amazon States Language (ASL). Untuk mempelajari selengkapnya, lihat Bekerja dengan layanan yang lain dan Meneruskan parameter ke API layanan.

Bagaimana integrasi Amazon SQS yang Dioptimalkan berbeda dari integrasi Amazon AWS SQS SDK

Tidak ada pengoptimalan untuk pola Minta Tanggapan atau Tunggu Panggilan Balik dengan Token Tugas integrasi.

API Amazon SQS yang didukung:

Note

Ada kuota untuk input maksimum atau ukuran data hasil untuk tugas di Step Functions. Ini membatasi Anda untuk 256 KB data sebagai string UTF-8 yang dikodekan ketika Anda mengirim ke, atau menerima data dari, layanan lain. Lihat Kuota yang berkaitan dengan eksekusi mesin status.

- `SendMessage`

  Parameter yang didukung:

  - `DelaySeconds`
  - `MessageAttribute`
  - `MessageBody`
  - `MessageDeduplicationId`
  - `MessageGroupId`
  - `QueueUrl`
- `Response syntax`

Parameter Step Functions dalam dinyatakan dalam PascalCase

Bahkan jika API layanan asli ada di camelCase, misalnya `startSyncExecution` tindakan API, Anda menentukan parameter PascalCase, seperti:. `StateMachineArn`

Berikut ini mencakup status `Task` yang mengirimkan pesan Amazon Simple Queue Service (Amazon SQS).

```
{
 "StartAt": "Send to SQS",
 "States": {
   "Send to SQS": {
     "Type": "Task",
     "Resource": "arn:aws:states:::sqs:sendMessage",
     "Parameters": {
       "QueueUrl": "https://sqs.us-east-1.amazonaws.com/123456789012/myQueue",
       "MessageBody.$": "$.input.message",
       "MessageAttributes": {
         "my_attribute_no_1": {
           "DataType": "String",
           "StringValue": "attribute1"
         },
         "my_attribute_no_2": {
           "DataType": "String",
           "StringValue": "attribute2"
         }
       }
     },
     "End": true
    }
  }
}
```

Berikut ini mencakup status `Task` yang menerbitkan ke antrean Amazon SQS, dan kemudian menunggu token tugas dikembalikan. Lihat Tunggu Panggilan Balik dengan Token Tugas.

```
{
   "StartAt":"Send message to SQS",
   "States":{
      "Send message to SQS":{
         "Type":"Task",
         "Resource":"arn:aws:states:::sqs:sendMessage.waitForTaskToken",
         "Parameters":{
            "QueueUrl":"https://sqs.us-east-1.amazonaws.com/123456789012/myQueue",
            "MessageBody":{
               "Input.$":"$",
               "TaskToken.$":"$$.Task.Token"
            }
         },
         "End":true
      }
```

```
    }
}
```

Untuk mempelajari selengkapnya tentang menerima pesan di Amazon SQS, lihat Terima dan Hapus Pesan Anda di Panduan Developer Amazon Simple Queue Service.

Untuk informasi tentang cara mengonfigurasi IAM izin saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Kelola AWS Step Functions Eksekusi sebagai Layanan Terpadu

Step Functions berintegrasi dengan API-nya sendiri sebagai integrasi layanan. Hal ini mengizinkan Step Functions untuk memulai eksekusi baru dari mesin status langsung dari status tugas eksekusi yang berjalan. Saat membangun alur kerja baru, gunakan eksekusi alur kerja yang di-nest untuk mengurangi kompleksitas alur kerja utama Anda dan untuk menggunakan kembali proses umum.

> Bagaimana integrasi Step Functions yang Dioptimalkan berbeda dari integrasi Step Functions AWS SDK
>
> - Pola Jalankan Tugas (.sync) integrasi tersedia.
>
> Perhatikan bahwa tidak ada pengoptimalan untuk pola Minta Tanggapan atau Tunggu Panggilan Balik dengan Token Tugas integrasi.

Untuk informasi selengkapnya, lihat hal berikut:

- Mulai Eksekusi dari Tugas
- Bekerja dengan layanan yang lain
- Meneruskan parameter ke API layanan

API dan sintaksis Step Functions yang didukung:

- `StartExecution`
  - Permintaan Sintaks
  - Parameter yang Didukung
    - `Input`

- Name
- StateMachineArn
- Sintaks respons

Berikut ini mencakup status Task yang memulai eksekusi mesin status lain dan menunggunya selesai.

```
{
   "Type":"Task",
   "Resource":"arn:aws:states:::states:startExecution.sync:2",
   "Parameters":{
      "Input":{
        "Comment": "Hello world!"
       },
      "StateMachineArn":"arn:aws:states:us-
east-1:123456789012:stateMachine:HelloWorld",
      "Name":"ExecutionName"
   },
   "End":true
}
```

Berikut ini mencakup status Task yang memulai eksekusi mesin status lain.

```
{
   "Type":"Task",
   "Resource":"arn:aws:states:::states:startExecution",
   "Parameters":{
      "Input":{
        "Comment": "Hello world!"
       },
      "StateMachineArn":"arn:aws:states:us-
east-1:123456789012:stateMachine:HelloWorld",
      "Name":"ExecutionName"
   },
   "End":true
}
```

Berikut ini mencakup status Task yang mengimplementasikan pola integrasi layanan callback.

```
{
```

```
   "Type":"Task",
   "Resource":"arn:aws:states:::states:startExecution.waitForTaskToken",
   "Parameters":{
      "Input":{
        "Comment": "Hello world!",
        "token.$": "$$.Task.Token"
       },
      "StateMachineArn":"arn:aws:states:us-
east-1:123456789012:stateMachine:HelloWorld",
      "Name":"ExecutionName"
   },
   "End":true
}
```

Untuk mengaitkan eksekusi alur kerja yang di-nest dengan eksekusi induk yang memulainya, teruskan parameter khusus bernama yang mencakup ID eksekusi yang ditarik dari objek konteks. Ketika memulai eksekusi yang di-nest, gunakan parameter bernama `AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID`. Teruskan ID eksekusi dengan menambahkan `.$` ke nama parameter, dan referensi ID dalam objek konteks dengan `$$.Execution.Id`. Untuk informasi selengkapnya, lihat Mengakses Obyek Konteks.

```
{
   "Type":"Task",
   "Resource":"arn:aws:states:::states:startExecution.sync",
   "Parameters":{
      "Input":{
        "Comment": "Hello world!",
        "AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID.$": "$$.Execution.Id"
       },
      "StateMachineArn":"arn:aws:states:us-
east-1:123456789012:stateMachine:HelloWorld",
      "Name":"ExecutionName"
   },
   "End":true
}
```

Mesin status yang di-nest mengembalikan hal berikut:

Sumber daya	Output
startExecution.sync	String
startExecution.sync:2	JSON

Keduanya akan menunggu mesin status yang di-nest selesai, tetapi mengembalikan format `Output` yang berbeda. Misalnya, jika Anda membuat fungsi Lambda yang mengembalikan objek `{ "MyKey": "MyValue" }`, Anda akan mendapatkan respons berikut:

Untuk startExecution.sync:

```
{
   <other fields>
   "Output": "{ \"MyKey\": \"MyValue\" }"
}
```

Untuk startExecution.sync:2:

```
{
   <other fields>
   "Output": {
      "MyKey": "MyValue"
   }
}
```

Mengkonfigurasi izin IAM untuk mesin status bersarang

Mesin status induk menentukan apakah mesin status anak telah menyelesaikan eksekusi menggunakan polling dan peristiwa. Polling memerlukan izin untuk `states:DescribeExecution` sementara peristiwa yang dikirim EventBridge ke Step Functions memerlukan izin untuk`events:PutTargets`,`events:PutRule`, dan. `events:DescribeRule` Jika izin ini hilang dari peran IAM Anda, mungkin ada penundaan sebelum mesin status induk mengetahui penyelesaian eksekusi mesin status anak.

Untuk mesin status yang memanggil `StartExecution` untuk eksekusi alur kerja bersarang tunggal, gunakan kebijakan IAM yang membatasi izin ke mesin status tersebut.

Untuk informasi selengkapnya, lihat izin IAM untuk Step Functions.

Panggil API pihak ketiga

Tugas HTTP adalah jenis Tugas status yang memungkinkan Anda memanggil API publik, pihak ketiga, seperti Salesforce dan Stripe, dalam alur kerja Anda. Untuk memanggil API pihak ketiga, gunakan status Tugas dengan `arn:aws:states:::http:invoke` sumber daya. Kemudian, berikan detail konfigurasi titik akhir API, seperti URL API, metode yang ingin Anda gunakan, dan detail otentikasi.

Jika Anda menggunakan Workflow Studio untuk membangun mesin status yang berisi Tugas HTTP, Workflow Studio akan secara otomatis menghasilkan peran eksekusi dengan IAM kebijakan untuk Tugas HTTP. Untuk informasi selengkapnya, lihat Peran untuk menguji Tugas HTTP di Workflow Studio.

Topik

- Definisi Tugas HTTP
- Bidang Tugas HTTP
- Otentikasi untuk Tugas HTTP
- Menggabungkan EventBridge koneksi dan data definisi Tugas HTTP
- Menerapkan pengkodean URL pada badan permintaan
- Izin IAM untuk menjalankan Tugas HTTP
- Contoh Tugas HTTP
- Menguji Tugas HTTP
- Respons HTTP Task yang tidak didukung

Definisi Tugas HTTP

Definisi ASL mewakili Tugas HTTP dengan `http:invoke` sumber daya. Definisi HTTP Task berikut memanggil Stripe API yang mengembalikan daftar semua pelanggan.

```
"Call third-party API": {
  "Type": "Task",
  "Resource": "arn:aws:states:::http:invoke",
  "Parameters": {
    "ApiEndpoint": "https://api.stripe.com/v1/customers",
    "Authentication": {
      "ConnectionArn": "arn:aws:events:us-east-2:123456789012:connection/
Stripe/81210c42-8af1-456b-9c4a-6ff02fc664ac"
```

```
    },
    "Method": "GET"
  },
  "End": true
}
```

Bidang Tugas HTTP

Tugas HTTP mencakup bidang-bidang berikut dalam definisinya.

`Resource` (Wajib)

Untuk menentukan jenis tugas, berikan ARN di lapangan. `Resource` Untuk Tugas HTTP, Anda menentukan `Resource` bidang sebagai berikut.

```
"Resource": "arn:aws:states:::http:invoke"
```

`Parameters` (Wajib)

Berisi`ApiEndpoint`,`Method`, dan `ConnectionArn` bidang yang menyediakan informasi tentang API pihak ketiga yang ingin Anda panggil. `Parameters`juga berisi bidang opsional, seperti `Headers` dan`QueryParameters`.

Anda dapat menentukan kombinasi JSON statis dan JsonPathsintaks seperti `Parameters` di lapangan. `Parameters` Untuk informasi selengkapnya, lihat Meneruskan parameter ke API layanan.

`ApiEndpoint`(Diperlukan)

Menentukan URL API pihak ketiga yang ingin Anda panggil. Untuk menambahkan parameter kueri ke URL, gunakan bidang. `QueryParameters` Contoh berikut menunjukkan bagaimana Anda dapat memanggil Stripe API untuk mengambil daftar semua pelanggan.

```
"ApiEndpoint":"https://api.stripe.com/v1/customers"
```

Anda juga dapat menentukan jalur referensi menggunakan JsonPathsintaks untuk memilih node JSON yang berisi URL API pihak ketiga. Misalnya, Anda ingin memanggil salah satu API Stripe menggunakan ID pelanggan tertentu. Bayangkan Anda telah memberikan masukan status berikut.

```
{
```

```
    "customer_id": "1234567890",
    "name": "John Doe"
}
```

Untuk mengambil detail ID pelanggan ini menggunakan Stripe API, tentukan `ApiEndpoint` seperti yang ditunjukkan pada contoh berikut. Contoh ini menggunakan fungsi intrinsik dan jalur referensi.

```
"ApiEndpoint.$":"States.Format('https://api.stripe.com/v1/customers/{}',
 $.customer_id)"
```

Saat runtime, Step Functions menyelesaikan nilai sebagai berikut. `ApiEndpoint`

```
https://api.stripe.com/v1/customers/1234567890
```

`Method`(Diperlukan)

Menentukan metode HTTP yang ingin Anda gunakan untuk memanggil API pihak ketiga. Anda dapat menentukan salah satu metode ini dalam Tugas HTTP Anda: GET, POST, PUT, DELETE, PATCH, OPTIONS, atau HEAD.

Misalnya, untuk menggunakan metode GET, tentukan `Method` bidang sebagai berikut.

```
"Method": "GET"
```

Anda juga dapat menggunakan jalur referensi untuk menentukan metode saat runtime. Misalnya, **`"Method.$": "$.myHTTPMethod"`**.

`Authentication`(Diperlukan)

Berisi `ConnectionArn` bidang yang menentukan cara mengautentikasi panggilan API pihak ketiga. Step Functionsmendukung otentikasi untuk yang ditentukan `ApiEndpoint` menggunakan sumber daya koneksi dari. Amazon EventBridge

`ConnectionArn`(Diperlukan)

Menentukan EventBridge koneksi ARN.

Tugas HTTP memerlukan EventBridge koneksi, yang mengelola kredensi otentikasi penyedia API dengan aman. Koneksi menentukan jenis otorisasi dan kredensi yang

akan digunakan untuk mengotorisasi API pihak ketiga. Menggunakan koneksi membantu Anda menghindari rahasia hard-coding, seperti kunci API, ke dalam definisi mesin status Anda. Dalam koneksi, Anda juga dapat menentukan`Headers`,`QueryParameters`, dan `RequestBody` parameter.

Saat Anda membuat EventBridge koneksi, Anda memberikan detail otentikasi Anda. Untuk informasi selengkapnya tentang cara kerja otentikasi untuk Tugas HTTP, lihatOtentikasi untuk Tugas HTTP.

Contoh berikut menunjukkan bagaimana Anda dapat menentukan `Authentication` bidang dalam definisi Tugas HTTP Anda.

```
"Authentication": {
  "ConnectionArn": "arn:aws:events:us-east-2:123456789012:connection/
Stripe/81210c42-8af1-456b-9c4a-6ff02fc664ac"
}
```

`Headers` (Opsional)

Menyediakan konteks dan metadata tambahan ke titik akhir API. Anda dapat menentukan header sebagai string atau array JSON.

Anda dapat menentukan header dalam EventBridge koneksi dan `Headers` bidang dalam Tugas HTTP. Kami menyarankan agar Anda tidak menyertakan detail otentikasi ke penyedia API Anda di `Headers` bidang ini. Kami menyarankan Anda memasukkan detail ini ke dalam EventBridge koneksi Anda.

Step Functionsmenambahkan header yang Anda tentukan dalam EventBridge koneksi ke header yang Anda tentukan dalam definisi Tugas HTTP. Jika tombol header yang sama hadir dalam definisi dan koneksi, Step Functions gunakan nilai yang sesuai yang ditentukan dalam EventBridge koneksi untuk header tersebut. Untuk informasi selengkapnya tentang cara Step Functions melakukan penggabungan data, lihatMenggabungkan EventBridge koneksi dan data definisi Tugas HTTP.

Contoh berikut menentukan header yang akan disertakan dalam panggilan API pihak ketiga:`content-type`.

```
"Headers": {
  "content-type": "application/json"
```

```
}
```

Anda juga dapat menggunakan jalur referensi untuk menentukan header saat runtime. Misalnya, **`"Headers.$": "$.myHTTPHeaders"`**.

Step FunctionsmenetapkanUser-Agent,Range, dan Host header. Step Functionsmenetapkan nilai Host header berdasarkan API yang Anda panggil. Berikut ini adalah contoh dari header ini.

```
User-Agent: Amazon|StepFunctions|HttpInvoke|us-east-1,
Range: bytes=0-262144,
Host: api.stripe.com
```

Anda tidak dapat menggunakan header berikut dalam definisi Tugas HTTP Anda. Jika Anda menggunakan header ini, Tugas HTTP gagal dengan States.Runtime kesalahan.

- A-IM
- Terima-Charset
- Terima-Datetime
- Terima-Pengenkodean
- Cache-Control
- Koneksi
- Pengkodean Konten
- Content-MD5
- Tanggal
- Harapkan
- Diteruskan
- Dari
- Host
- HTTP2-pengaturan
- Jika-Pertandingan
- Jika-Modifikasi-Sejak
- Jika-Tidak Pertandingan
- Jika-Rentang

- Jika-Tidak Dimodifikasi-Sejak
- Max-Maju
- Asal
- Pragma
- Otorisasi Proksi
- Perujuk
- Server
- TE
- Trailer
- Transfer-Encoding
- Peningkatan
- Melalui
- Peringatan
- x-diteruskan-*
- x-amz-*
- x-amzn-*

`QueryParameters` (Opsional)

Menyisipkan pasangan nilai kunci di akhir URL API. Anda dapat menentukan parameter kueri sebagai string, array JSON, atau objek JSON. Step Functionssecara otomatis mengkodekan parameter kueri URL saat memanggil API pihak ketiga.

Misalnya, katakan bahwa Anda ingin memanggil Stripe API untuk mencari pelanggan yang melakukan transaksi mereka dalam dolar AS (USD). Bayangkan Anda telah memberikan yang berikut ini `QueryParameters` sebagai input status.

```
"QueryParameters": {
  "currency": "usd"
}
```

Saat runtime, Step Functions menambahkan `QueryParameters` ke URL API sebagai berikut.

```
https://api.stripe.com/v1/customers/search?currency=usd
```

Anda juga dapat menggunakan jalur referensi untuk menentukan parameter kueri saat runtime. Misalnya, **`"QueryParameters.$": "$.myQueryParameters"`**.

Jika Anda telah menentukan parameter kueri dalam EventBridge koneksi Anda, Step Functions tambahkan parameter kueri ini ke parameter kueri yang Anda tentukan dalam definisi Tugas HTTP. Jika kunci parameter kueri yang sama ada dalam definisi dan koneksi, Step Functions gunakan nilai yang sesuai yang ditentukan dalam EventBridge koneksi untuk header tersebut. Untuk informasi selengkapnya tentang cara Step Functions melakukan penggabungan data, lihatMenggabungkan EventBridge koneksi dan data definisi Tugas HTTP.

`Transform` (Opsional)

Berisi `RequestBodyEncoding` dan `RequestEncodingOptions` bidang. Secara default, Step Functions mengirimkan badan permintaan sebagai data JSON ke titik akhir API.

Jika penyedia API Anda menerima badan `form-urlencoded` permintaan, gunakan `Transform` bidang untuk menentukan pengkodean URL untuk badan permintaan. Anda juga harus menentukan `content-type` header sebagai`application/x-www-form-urlencoded`. Step Functionskemudian secara otomatis URL-menyandikan badan permintaan Anda.

`RequestBodyEncoding`

Menentukan URL-encoding dari badan permintaan Anda. Anda dapat menentukan satu nilai ini: `NONE` atau`URL_ENCODED`.

- `NONE`— Badan permintaan HTTP akan menjadi JSON serialisasi lapangan. `RequestBody` Ini adalah nilai default.
- `URL_ENCODED`— Badan permintaan HTTP akan menjadi data formulir yang dikodekan URL dari bidang tersebut. `RequestBody`

`RequestEncodingOptions`

Menentukan opsi pengkodean yang akan digunakan untuk array di badan permintaan Anda jika Anda menyetel `RequestBodyEncoding` ke. `URL_ENCODED`

Step Functionsmendukung opsi pengkodean array berikut. Untuk informasi lebih lanjut tentang opsi ini dan contohnya, lihatMenerapkan pengkodean URL pada badan permintaan.

- `INDICES`— Mengkodekan array menggunakan nilai indeks elemen array. Secara default, Step Functions gunakan opsi pengkodean ini.

- `REPEAT`— Mengulangi kunci untuk setiap item dalam array.
- `COMMAS`— Mengkodekan semua nilai dalam kunci sebagai daftar nilai yang dibatasi koma.
- `BRACKETS`— Mengulangi kunci untuk setiap item dalam array dan menambahkan braket, [], ke kunci untuk menunjukkan bahwa itu adalah array.

Contoh berikut menetapkan URL-encoding untuk data badan permintaan. Ini juga menentukan untuk menggunakan opsi `COMMAS` pengkodean untuk array di badan permintaan.

```
"Transform": {
  "RequestBodyEncoding": "URL_ENCODED",
  "RequestEncodingOptions": {
    "ArrayFormat": "COMMAS"
  }
}
```

`RequestBody` (Opsional)

Menerima data JSON yang Anda berikan dalam input status. Di`RequestBody`, Anda dapat menentukan kombinasi JSON statis dan JsonPathsintaks. Misalnya, katakan bahwa Anda memberikan masukan status berikut:

```
{
    "CardNumber": "1234567890",
    "ExpiryDate": "09/25"
}
```

Untuk menggunakan nilai-nilai ini dari `CardNumber` dan `ExpiryDate` dalam badan permintaan Anda saat runtime, Anda dapat menentukan data JSON berikut di badan permintaan Anda.

```
"RequestBody": {
  "Card": {
    "Number.$": "$.CardNumber",
    "Expiry.$": "$.ExpiryDate",
    "Name": "John Doe",
    "Address": "123 Any Street, Any Town, USA"
  }
}
```

Jika API pihak ketiga yang ingin Anda panggil memerlukan badan `form-urlencoded` permintaan, Anda harus menentukan pengkodean URL untuk data isi permintaan Anda. Untuk informasi selengkapnya, lihat Menerapkan pengkodean URL pada badan permintaan.

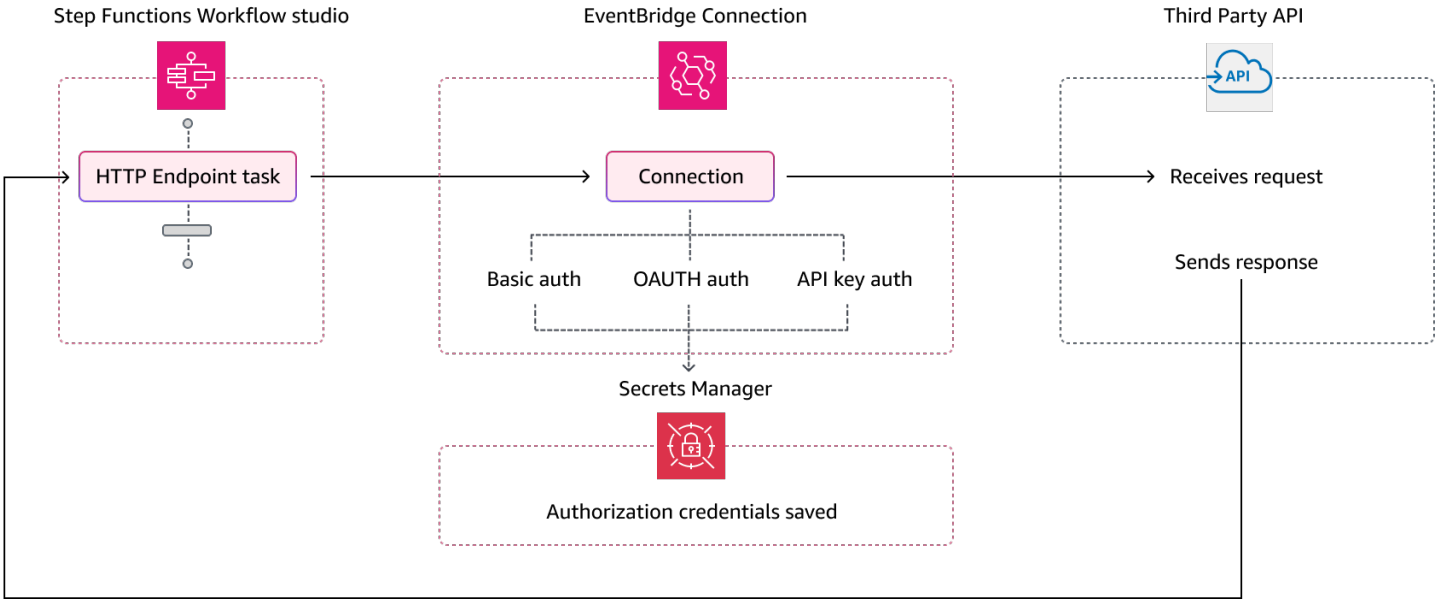
Otentikasi untuk Tugas HTTP

Tugas HTTP memerlukan EventBridge koneksi, yang mengelola kredensi otentikasi penyedia API dengan aman. Koneksi menentukan jenis otorisasi dan kredensi yang akan digunakan untuk mengotorisasi API pihak ketiga. Menggunakan koneksi membantu Anda menghindari rahasia hard-coding, seperti kunci API, ke dalam definisi mesin status Anda. EventBridge Koneksi mendukung skema otorisasi Basic, OAuth, dan API Key.

Saat Anda membuat EventBridge koneksi, Anda memberikan detail otentikasi Anda. Anda juga dapat menyertakan parameter header, body, dan query yang diperlukan untuk otorisasi dengan API. Anda harus menyertakan ARN koneksi dalam Tugas HTTP apa pun yang memanggil API pihak ketiga.

Saat Anda membuat koneksi dan menambahkan parameter otorisasi, EventBridge buat rahasia di AWS Secrets Manager. Dalam rahasia ini, EventBridge menyimpan parameter koneksi dan otorisasi dalam bentuk terenkripsi. Agar berhasil membuat atau memperbarui koneksi, Anda harus menggunakan Secrets Manager Akun AWS yang memiliki izin untuk menggunakan Secrets Manager. Untuk informasi selengkapnya tentang IAM izin yang dibutuhkan mesin status Anda untuk mengakses EventBridge sambungan, lihatIzin IAM untuk menjalankan Tugas HTTP.

Gambar berikut menunjukkan cara Step Functions menangani otentikasi untuk panggilan API pihak ketiga menggunakan EventBridge koneksi.



Menggabungkan EventBridge koneksi dan data definisi Tugas HTTP

Saat Anda menjalankan Tugas HTTP, Anda dapat menentukan data dalam EventBridge koneksi dan definisi Tugas HTTP Anda. Data ini mencakup`Headers`,`QueryParameters`, dan `RequestBody` parameter. Sebelum memanggil API pihak ketiga, Step Functions menggabungkan badan permintaan dengan parameter badan koneksi dalam semua kasus kecuali jika badan permintaan Anda adalah string dan parameter badan koneksi tidak kosong. Dalam hal ini, Tugas HTTP gagal dengan `States.Runtime` kesalahan.

Jika ada kunci duplikat yang ditentukan dalam definisi Tugas HTTP dan EventBridge koneksi, Step Functions timpa nilai dalam Tugas HTTP dengan nilai-nilai dalam koneksi.

Daftar berikut menjelaskan cara Step Functions menggabungkan data sebelum memanggil API pihak ketiga:

- Header - Step Functions menambahkan header apa pun yang Anda tentukan dalam koneksi ke header di `Headers` bidang Tugas HTTP. Jika ada konflik antara tombol header, Step Functions gunakan nilai yang ditentukan dalam koneksi untuk header tersebut. Misalnya, jika Anda menentukan `content-type` header dalam definisi HTTP Task dan EventBridge koneksi, Step Functions menggunakan nilai `content-type` header yang ditentukan dalam koneksi.
- Parameter kueri - Step Functions menambahkan parameter kueri apa pun yang Anda tentukan dalam koneksi ke parameter kueri di `QueryParameters` bidang Tugas HTTP. Jika ada konflik antara kunci parameter kueri, Step Functions gunakan nilai yang ditentukan dalam koneksi untuk parameter kueri tersebut. Misalnya, jika Anda menentukan parameter `maxItems` kueri dalam definisi HTTP Task dan EventBridge koneksi, Step Functions menggunakan nilai parameter `maxItems` kueri yang ditentukan dalam koneksi.
- Parameter tubuh
  - Step Functionsmenambahkan nilai badan permintaan apa pun yang ditentukan dalam koneksi ke badan permintaan di `RequestBody` bidang Tugas HTTP. Jika ada konflik antara kunci badan permintaan, Step Functions gunakan nilai yang ditentukan dalam koneksi untuk badan permintaan. Misalnya, katakan bahwa Anda menentukan `Mode` bidang dalam `RequestBody` definisi HTTP Task dan EventBridge koneksi. Step Functionsmenggunakan nilai `Mode` bidang yang Anda tentukan dalam koneksi.
  - Jika Anda menentukan badan permintaan sebagai string alih-alih objek JSON, dan EventBridge koneksi juga berisi badan permintaan, tidak Step Functions dapat menggabungkan badan permintaan yang ditentukan di kedua tempat ini. Gagal Tugas HTTP dengan `States.Runtime` kesalahan.

Step Functionsmenerapkan semua transformasi dan membuat serial badan permintaan setelah menyelesaikan penggabungan badan permintaan.

Contoh berikut menetapkan`Headers`,`QueryParameters`, dan `RequestBody` bidang di kedua Tugas HTTP dan EventBridge koneksi.

Definisi Tugas HTTP

```
{
  "Comment": "Data merging example for HTTP Task and EventBridge connection",
  "StartAt": "ListCustomers",
  "States": {
    "ListCustomers": {
      "Type": "Task",
      "Resource": "arn:aws:states:::http:invoke",
      "Parameters": {
        "Authentication": {
          "ConnectionArn": "arn:aws:events:us-
east-1:123456789012:connection/Example/81210c42-8af1-456b-9c4a-6ff02fc664ac"
        },
        "ApiEndpoint": "https:/example.com/path",
        "Method": "GET",
        "Headers": {
          "Request-Id": "my_request_id",
          "Header-Param": "state_machine_header_param"
        },
        "RequestBody": {
          "Job": "Software Engineer",
          "Company": "AnyCompany",
          "BodyParam": "state_machine_body_param"
        },
        "QueryParameters": {
          "QueryParam": "state_machine_query_param"
        }
      }
    }
  }
}
```

EventBridgekoneksi

```
{
  "AuthorizationType": "API_KEY",
  "AuthParameters": {
    "ApiKeyAuthParameters": {
      "ApiKeyName": "ApiKey",
      "ApiKeyValue": "key_value"
    },
    "InvocationHttpParameters": {
      "BodyParameters": [
        {
          "Key": "BodyParam",
          "Value": "connection_body_param"
        }
      ],
      "HeaderParameters": [
        {
          "Key": "Header-Param",
          "Value": "connection_header_param"
        }
      ],
      "QueryStringParameters": [
        {
          "Key": "QueryParam",
          "Value": "connection_query_param"
        }
      ]
    }
  }
}
```

Dalam contoh ini, kunci duplikat ditentukan dalam Tugas HTTP dan EventBridge koneksi. Oleh karena itu, Step Functions menimpa nilai-nilai dalam Tugas HTTP dengan nilai-nilai dalam koneksi. Cuplikan kode berikut menunjukkan permintaan HTTP yang Step Functions dikirim ke API pihak ketiga.

```
POST /path?QueryParam=connection_query_param HTTP/1.1
Apikey: key_value
Content-Length: 79
Content-Type: application/json; charset=UTF-8
Header-Param: connection_header_param
Host: example.com
Range: bytes=0-262144
```

```
Request-Id: my_request_id
User-Agent: Amazon|StepFunctions|HttpInvoke|us-east-1

{"Job":"Software Engineer","Company":"AnyCompany","BodyParam":"connection_body_param"}
```

Menerapkan pengkodean URL pada badan permintaan

Secara default, Step Functions mengirimkan badan permintaan sebagai data JSON ke titik akhir API. Jika penyedia API pihak ketiga Anda memerlukan badan `form-urlencoded` permintaan, Anda harus menentukan pengkodean URL untuk badan permintaan. Step Functionskemudian secara otomatis URL-menyandikan badan permintaan berdasarkan opsi pengkodean URL yang Anda pilih.

Anda menentukan pengkodean URL menggunakan bidang. `Transform` Bidang ini berisi `RequestBodyEncoding` bidang yang menentukan apakah Anda ingin menerapkan pengkodean URL untuk badan permintaan Anda atau tidak. Saat Anda menentukan `RequestBodyEncoding` bidang, Step Functions mengonversi badan permintaan JSON Anda menjadi badan `form-urlencoded` permintaan sebelum memanggil API pihak ketiga. Anda juga harus menentukan `content-type` header sebagai `application/x-www-form-urlencoded` karena API yang menerima data yang disandikan URL mengharapkan header. `content-type`

Untuk menyandikan array di badan permintaan Anda, Step Functions berikan opsi pengkodean array berikut.

- `INDICES`— Mengulangi kunci untuk setiap item dalam array dan menambahkan braket, [], ke kunci untuk menunjukkan bahwa itu adalah array. Braket ini berisi indeks elemen array. Menambahkan indeks membantu Anda menentukan urutan elemen array. Secara default, Step Functions gunakan opsi pengkodean ini.

  Misalnya, jika badan permintaan Anda berisi array berikut.

  ```
  {"array": ["a","b","c","d"]}
  ```

  Step Functionsmengkodekan array ini ke string berikut.

  ```
  array[0]=a&array[1]=b&array[2]=c&array[3]=d
  ```

- `REPEAT`— Mengulangi kunci untuk setiap item dalam array.

  Misalnya, jika badan permintaan Anda berisi array berikut.

```
{"array": ["a","b","c","d"]}
```

Step Functionsmengkodekan array ini ke string berikut.

```
array=a&array=b&array=c&array=d
```

- `COMMAS`— Mengkodekan semua nilai dalam kunci sebagai daftar nilai yang dibatasi koma.

  Misalnya, jika badan permintaan Anda berisi array berikut.

  ```
  {"array": ["a","b","c","d"]}
  ```

  Step Functionsmengkodekan array ini ke string berikut.

  ```
  array=a,b,c,d
  ```

- `BRACKETS`— Mengulangi kunci untuk setiap item dalam array dan menambahkan braket, `[]`, ke kunci untuk menunjukkan bahwa itu adalah array.

  Misalnya, jika badan permintaan Anda berisi array berikut.

  ```
  {"array": ["a","b","c","d"]}
  ```

  Step Functionsmengkodekan array ini ke string berikut.

  ```
  array[]=a&array[]=b&array[]=c&array[]=d
  ```

Izin IAM untuk menjalankan Tugas HTTP

Peran eksekusi mesin status Anda harus memiliki`states:InvokeHTTPEndpoint`,`events:RetrieveConnectionCredentials`,`secretsmanag` dan `secretsmanager:DescribeSecret` izin untuk Tugas HTTP untuk memanggil API pihak ketiga. Contoh kebijakan IAM berikut memberikan hak istimewa paling sedikit yang diperlukan untuk peran mesin status Anda untuk memanggil Stripe API. Kebijakan IAM ini juga memberikan izin ke peran mesin status untuk mengakses EventBridge koneksi tertentu, termasuk rahasia untuk koneksi ini yang disimpan di Secrets Manager.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Statement1",
            "Effect": "Allow",
            "Action": "states:InvokeHTTPEndpoint",
            "Resource": "arn:aws:states:us-
east-2:123456789012:stateMachine:myStateMachine",
            "Condition": {
                "StringEquals": {
                    "states:HTTPMethod": "GET"
                },
                "StringLike": {
                    "states:HTTPEndpoint": "https://api.stripe.com/*"
                }
            }
        },
        {
            "Sid": "Statement2",
            "Effect": "Allow",
            "Action": [
                "events:RetrieveConnectionCredentials",
            ],
            "Resource": "arn:aws:events:us-
east-2:123456789012:connection/oauth_connection/aeabd89e-d39c-4181-9486-9fe03e6f286a"
        },
        {
            "Sid": "Statement3",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret"
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:events!connection/*"
        }
    ]
}
```

Contoh Tugas HTTP

Definisi mesin status berikut menunjukkan Tugas HTTP yang mencakup`Headers`,, `QueryParametersTransform`, dan `RequestBody` parameter. Tugas HTTP memanggil Stripe API, https://api.stripe.com/v1/invoices, untuk menghasilkan faktur. Tugas HTTP juga menentukan URL-encoding untuk badan permintaan menggunakan opsi encoding. `INDICES`

Pastikan Anda telah membuat EventBridge koneksi. Contoh berikut menunjukkan koneksi yang dibuat menggunakan jenis autentikasi BASIC.

```
{
    "Type": "BASIC",
    "AuthParameters": {
        "BasicAuthParameters": {
            "Password": "myPassword",
            "Username": "myUsername"
         },
    }
}
```

Ingatlah untuk mengganti teks yang *dicetak miring dengan informasi* spesifik sumber daya Anda.

```
{
  "Comment": "A state machine that uses HTTP Task",
  "StartAt": "CreateInvoiceAPI",
  "States": {
    "CreateInvoiceAPI": {
      "Type": "Task",
      "Resource": "arn:aws:states:::http:invoke",
      "Parameters": {
        "ApiEndpoint": "https://api.stripe.com/v1/invoices",
        "Method": "POST",
        "Authentication": {
          "ConnectionArn": ""arn:aws:events:us-east-2:123456789012:connection/
Stripe/81210c42-8af1-456b-9c4a-6ff02fc664ac"
        },
        "Headers": {
          "Content-Type": "application/x-www-form-urlencoded"
        },
        "RequestBody": {
          "customer.$": "$.customer_id",
```

```
          "description": "Monthly subscription",
          "metadata": {
            "order_details": "monthly report data"
          }
        },
        "Transform": {
          "RequestBodyEncoding": "URL_ENCODED",
          "RequestEncodingOptions": {
            "ArrayFormat": "INDICES"
          }
        }
      },
      "Retry": [
        {
          "ErrorEquals": [
            "States.Http.StatusCode.429",
            "States.Http.StatusCode.503",
            "States.Http.StatusCode.504",
            "States.Http.StatusCode.502"
          ],
          "BackoffRate": 2,
          "IntervalSeconds": 1,
          "MaxAttempts": 3,
          "JitterStrategy": "FULL"
        }
      ],
      "Catch": [
        {
          "ErrorEquals": [
            "States.Http.StatusCode.404",
            "States.Http.StatusCode.400",
            "States.Http.StatusCode.401",
            "States.Http.StatusCode.409",
            "States.Http.StatusCode.500"
          ],
          "Comment": "Handle all non 200 ",
          "Next": "HandleInvoiceFailure"
        }
      ],
      "End": true
    }
  }
}
```

Untuk menjalankan mesin status ini, berikan ID pelanggan sebagai input seperti yang ditunjukkan pada contoh berikut:

```
{
    "customer_id": "1234567890"
}
```

Contoh berikut menunjukkan permintaan HTTP yang Step Functions mengirim ke Stripe API.

```
POST /v1/invoices HTTP/1.1
Authorization: Basic <base64 of username and password>
Content-Type: application/x-www-form-urlencoded
Host: api.stripe.com
Range: bytes=0-262144
Transfer-Encoding: chunked
User-Agent: Amazon|StepFunctions|HttpInvoke|us-east-1

description=Monthly%20subscription&metadata%5Border_details%5D=monthly%20report
%20data&customer=1234567890
```

Menguji Tugas HTTP

Anda dapat menggunakan TestStateAPI melalui konsol, SDK, atau AWS CLI untuk menguji Tugas HTTP. Prosedur berikut menjelaskan cara menggunakan TestState API di Step Functions konsol. Anda dapat menguji detail permintaan, respons, dan autentikasi API secara berulang hingga Tugas HTTP berfungsi seperti yang diharapkan.

Uji status Tugas HTTP di Step Functions konsol

1. Buka Konsol Step Functions.
2. Pilih Buat mesin status untuk mulai membuat mesin status atau pilih mesin status yang ada yang berisi Tugas HTTP.

   Lihat Langkah 4 jika Anda menguji tugas di mesin status yang ada.
3. Di Mode desain Workflow Studio, konfigurasikan Tugas HTTP secara visual. Atau pilih mode Kode untuk menyalin-menempelkan definisi mesin status dari lingkungan pengembangan lokal Anda.
4. Dalam mode Desain, pilih Status uji di Inspector panel Workflow Studio.
5. Dalam kotak dialog Test state, lakukan hal berikut:

a. Untuk peran Eksekusi, pilih peran eksekusi untuk menguji status. Jika Anda tidak memiliki peran dengan izin yang cukup untuk Tugas HTTP, lihat Peran untuk menguji Tugas HTTP di Workflow Studio untuk membuat peran.

b. (Opsional) Berikan masukan JSON apa pun yang dibutuhkan status yang Anda pilih untuk pengujian.

c. Untuk tingkat Inspeksi, pertahankan pilihan default INFO. Level ini menunjukkan status panggilan API dan output status. Ini berguna untuk memeriksa respons API dengan cepat.

d. Pilih Mulai tes.

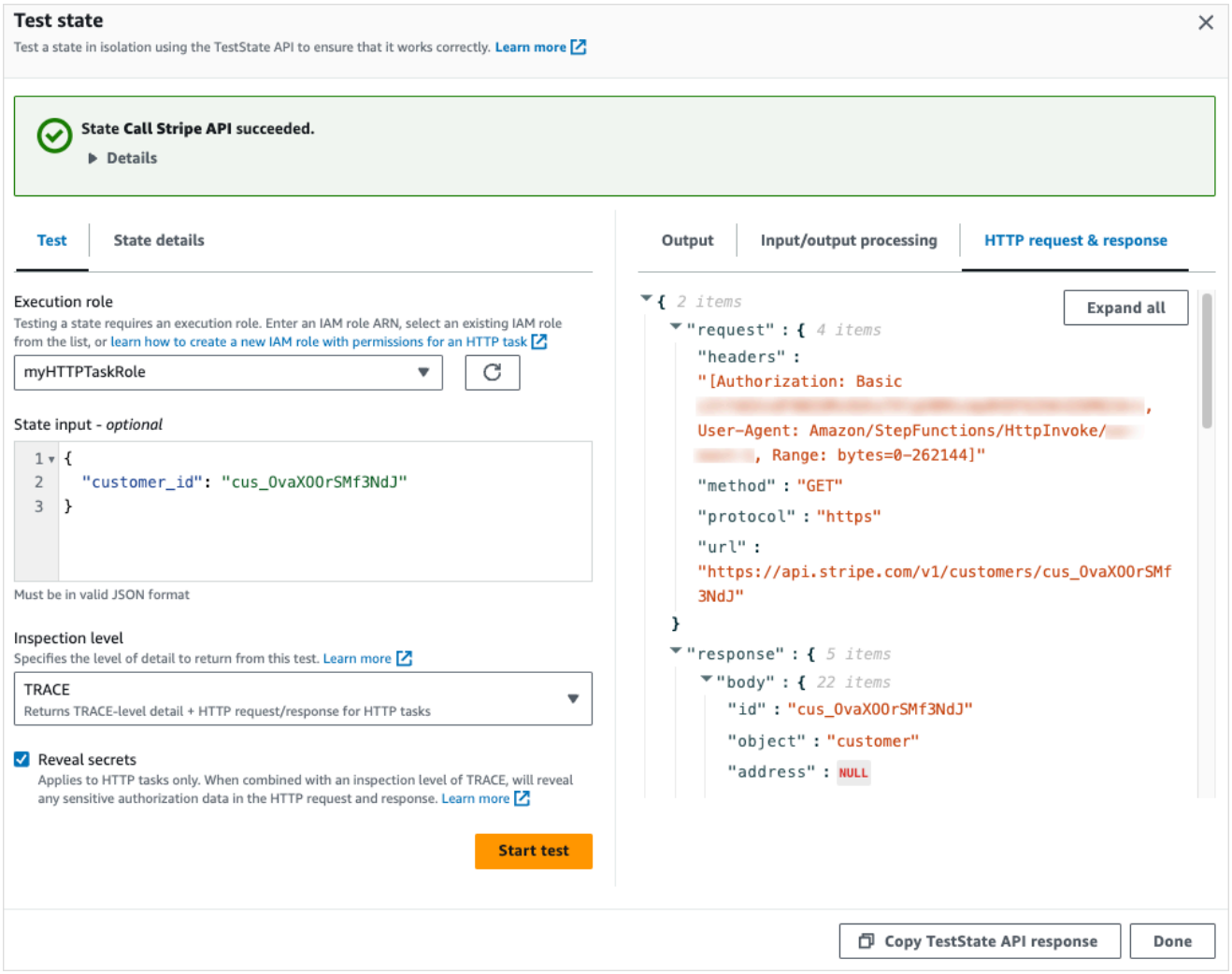
e. Jika tes berhasil, output status muncul di sisi kanan kotak dialog status Uji. Jika tes gagal, kesalahan muncul.

   Di tab Detail negara pada kotak dialog, Anda dapat melihat definisi status dan tautan ke EventBridgekoneksi Anda.

f. Ubah level Inspeksi menjadi TRACE. Level ini menunjukkan permintaan dan respons HTTP mentah, dan berguna untuk memverifikasi header, parameter kueri, dan detail spesifik API lainnya.

g. Pilih kotak centang Reveal Secrets. Dalam kombinasi dengan TRACE, pengaturan ini memungkinkan Anda melihat data sensitif yang disisipkan EventBridge koneksi, seperti kunci API. Identitas IAM pengguna yang Anda gunakan untuk mengakses konsol harus memiliki izin untuk melakukan `states:RevealSecrets` tindakan. Tanpa izin ini, Step Functions melempar kesalahan akses ditolak saat Anda memulai pengujian. Untuk contoh IAM kebijakan yang menetapkan `states:RevealSecrets` izin, lihatIAMizin untuk menggunakan API TestState .

   Gambar berikut menunjukkan tes untuk Tugas HTTP yang berhasil. Tingkat Inspeksi untuk status ini diatur ke TRACE. Tab permintaan & respons HTTP pada gambar berikut menunjukkan hasil panggilan API pihak ketiga.



h. Pilih Mulai tes.

i. Jika pengujian berhasil, Anda dapat melihat detail HTTP Anda di bawah tab permintaan & respons HTTP.

Respons HTTP Task yang tidak didukung

Tugas HTTP gagal dengan `States.Runtime` kesalahan jika salah satu kondisi berikut benar untuk respons yang dikembalikan:

- Respons berisi header tipe konten`application/octet-stream`,, `image/*``video/*`, atau. `audio/*`
- Respons tidak dapat dibaca sebagai string yang valid. Misalnya, data biner atau gambar.

Pola integrasi layanan

AWS Step Functions terintegrasi dengan layanan langsung di Amazon States Language. Anda dapat mengendalikan layanan AWS ini menggunakan tiga pola integrasi layanan:

- Memanggil layanan dan memungkinkan Step Functions maju ke status berikutnya segera setelah mendapat respons HTTP.
- Memanggil layanan dan membuat Step Functions menunggu tugas untuk diselesaikan.
- Memanggil layanan dengan token tugas dan membuat Step Functions menunggu sampai token tersebut dikembalikan dengan muatan.

Masing-masing pola integrasi layanan ini dikendalikan oleh cara Anda membuat URI di bidang `"Resource"` ketentuan tugas Anda.

Cara Memanggil Layanan Terintegrasi

- Minta Tanggapan
- Jalankan Tugas (.sync)
- Tunggu Panggilan Balik dengan Token Tugas

Untuk informasi tentang konfigurasi AWS Identity and Access Management (IAM) untuk layanan terintegrasi, lihat. Kebijakan IAM untuk layanan terintegrasi

Minta Tanggapan

Ketika Anda menentukan layanan di string `"Resource"` status tugas Anda, dan Anda hanya menyediakan sumber daya, Step Functions akan menunggu respons HTTP lalu maju ke status berikutnya. Step Functions tidak akan menunggu tugas selesai.

Contoh berikut menunjukkan cara Anda dapat memublikasikan topik Amazon SNS.

```
"Send message to SNS":{
   "Type":"Task",
   "Resource":"arn:aws:states:::sns:publish",
   "Parameters":{
      "TopicArn":"arn:aws:sns:us-east-1:123456789012:myTopic",
      "Message":"Hello from Step Functions!"
   },
```

```
   "Next":"NEXT_STATE"
}
```

Contoh ini mereferensikan API Publikasikan Amazon SNS. Alur kerja berlangsung ke status berikutnya setelah memanggil API `Publish`.

> **Tip**
>
> Untuk menerapkan alur kerja sampel yang menggunakan pola integrasi layanan Respons Permintaan ke Anda Akun AWS, lihat Modul 2 - Permintaan Respons Lokakarya. AWS Step Functions

Jalankan Tugas (.sync)

Untuk layanan terintegrasi seperti AWS Batch dan Amazon ECS, Step Functions dapat menunggu permintaan selesai sebelum melanjutkan ke status berikutnya. Untuk membuat Step Functions menunggu, tentukan bidang `"Resource"` di ketentuan status tugas Anda dengan sufiks `.sync` yang ditambahkan setelah URI sumber daya.

Misalnya, saat mengirimkan AWS Batch pekerjaan, gunakan `"Resource"` bidang dalam definisi mesin status seperti yang ditunjukkan dalam contoh ini.

```
"Manage Batch task": {
  "Type": "Task",
  "Resource": "arn:aws:states:::batch:submitJob.sync",
  "Parameters": {
    "JobDefinition": "arn:aws:batch:us-east-2:123456789012:job-definition/
testJobDefinition",
    "JobName": "testJob",
    "JobQueue": "arn:aws:batch:us-east-2:123456789012:job-queue/testQueue"
  },
  "Next": "NEXT_STATE"
}
```

Saat bagian `.sync` ditambahkan ke sumber daya Amazon Resource Name (ARN), ini berarti bahwa Step Functions menunggu tugas selesai. Setelah memanggil AWS Batch `submitJob`, alur kerja dijeda. Ketika tugas selesai, Step Functions maju ke status berikutnya. Untuk informasi lebih lanjut, lihat AWS Batch contoh proyek:Mengelola pekerjaan batch (AWS Batch,Amazon SNS).

Jika tugas menggunakan ini (`.sync`) pola integrasi layanan dibatalkan, dan Step Functions tidak dapat membatalkan tugas, Anda mungkin dikenakan biaya tambahan dari layanan terintegrasi. Sebuah tugas dapat dibatalkan jika:

- Eksekusi mesin status dihentikan.
- Cabang tatus Paralel yang berbeda gagal dengan kesalahan yang tidak tertangkap.
- Perulangan dari Status Peta gagal dengan kesalahan yang tidak tertangkap.

Step Functions akan membuat upaya terbaik untuk membatalkan tugas. Misalnya, jika tugas `states:startExecution.sync` Step Functions dibatalkan, tugas tersebut akan memanggil tindakan API `StopExecution` Step Functions. Namun, ada kemungkinan bahwa Step Functions akan dapat membatalkan tugas. Alasan tersebut termasuk, namun tidak terbatas pada:

- Peran eksekusi IAM Anda tidak memiliki izin untuk membuat panggilan API yang sesuai.
- Terjadi pemadaman layanan sementara.

Saat Anda menggunakan pola integrasi `.sync` layanan, Step Functions menggunakan polling yang menggunakan kuota dan peristiwa yang ditetapkan untuk memantau status pekerjaan. Untuk `.sync` pemanggilan dalam akun yang sama, Step Functions menggunakan EventBridge peristiwa dan melakukan polling API yang Anda tentukan dalam status. `Task` Untuk `.sync` pemanggilan lintas akun, Step Functions hanya menggunakan polling. Misalnya`states:StartExecution.sync`, Step Functions melakukan polling di DescribeExecutionAPI dan menggunakan kuota yang Anda tetapkan.

> **Tip**
>
> Untuk menerapkan alur kerja contoh yang menggunakan pola integrasi layanan Run a Job (.sync) ke Anda Akun AWS, lihat Modul 3 - Jalankan Job (.sync) dari Workshop. AWS Step Functions

Untuk melihat daftar dukungan layanan terintegrasi yang menunggu tugas selesai (`.sync`), lihat Integrasi yang dioptimalkan untuk Step Functions.

> **Note**
>
> Integrasi layanan yang menggunakan pola `.sync` memerlukan izin IAM tambahan. Untuk informasi selengkapnya, lihat Kebijakan IAM untuk layanan terintegrasi.

Dalam beberapa kasus, Anda mungkin ingin Step Functions melanjutkan alur kerja Anda sebelum pekerjaan selesai sepenuhnya. Anda dapat mencapai ini dengan cara yang sama seperti saat menggunakan pola integrasi Tunggu Panggilan Balik dengan Token Tugas layanan. Untuk melakukan ini, berikan token tugas ke pekerjaan Anda, lalu kembalikan menggunakan panggilan `SendTaskFailure`API `SendTaskSuccess`atau. Step Functions akan menggunakan data yang Anda berikan dalam panggilan tersebut untuk menyelesaikan tugas, berhenti memantau pekerjaan, dan melanjutkan alur kerja.

Tunggu Panggilan Balik dengan Token Tugas

Tugas panggilan balik menyediakan cara untuk menjeda alur kerja sampai token tugas dikembalikan. Tugas mungkin perlu menunggu persetujuan seseorang, mengintegrasikan dengan pihak ketiga, atau memanggil sistem warisan. Untuk tugas seperti ini, Anda dapat menjeda Step Functions hingga eksekusi alur kerja mencapai kuota layanan satu tahun (lihat,Kuota terkait throttling status), dan menunggu proses eksternal atau alur kerja selesai. Untuk situasi ini Step Functions memungkinkan Anda meneruskan token tugas ke integrasi layanan AWS SDK, dan juga ke beberapa integrasi layanan yang Dioptimalkan. Tugas akan terjeda sampai menerima token tugas yang dikembalikan dengan panggilan `SendTaskSuccess` atau `SendTaskFailure`.

Jika `Task` status yang menggunakan token tugas panggilan balik habis, token acak baru akan dihasilkan. Anda dapat mengakses token tugas dari objek konteks.

> **Note**
>
> Token tugas harus berisi setidaknya satu karakter, dan tidak dapat melebihi 1024 karakter.

Untuk menggunakan `.waitForTaskToken` integrasi AWS SDK, API yang Anda gunakan harus memiliki bidang parameter untuk menempatkan token tugas.

> **Note**
>
> Anda harus meneruskan token tugas dari kepala sekolah dalam akun yang sama. AWS Token tidak akan berfungsi jika Anda mengirimnya dari kepala sekolah di akun yang berbeda. AWS

> **Tip**
>
> Untuk menerapkan alur kerja sampel yang menggunakan pola integrasi layanan token tugas callback ke Anda Akun AWS, lihat Modul 4 - Tunggu Panggilan Balik dengan Token Tugas Lokakarya. AWS Step Functions

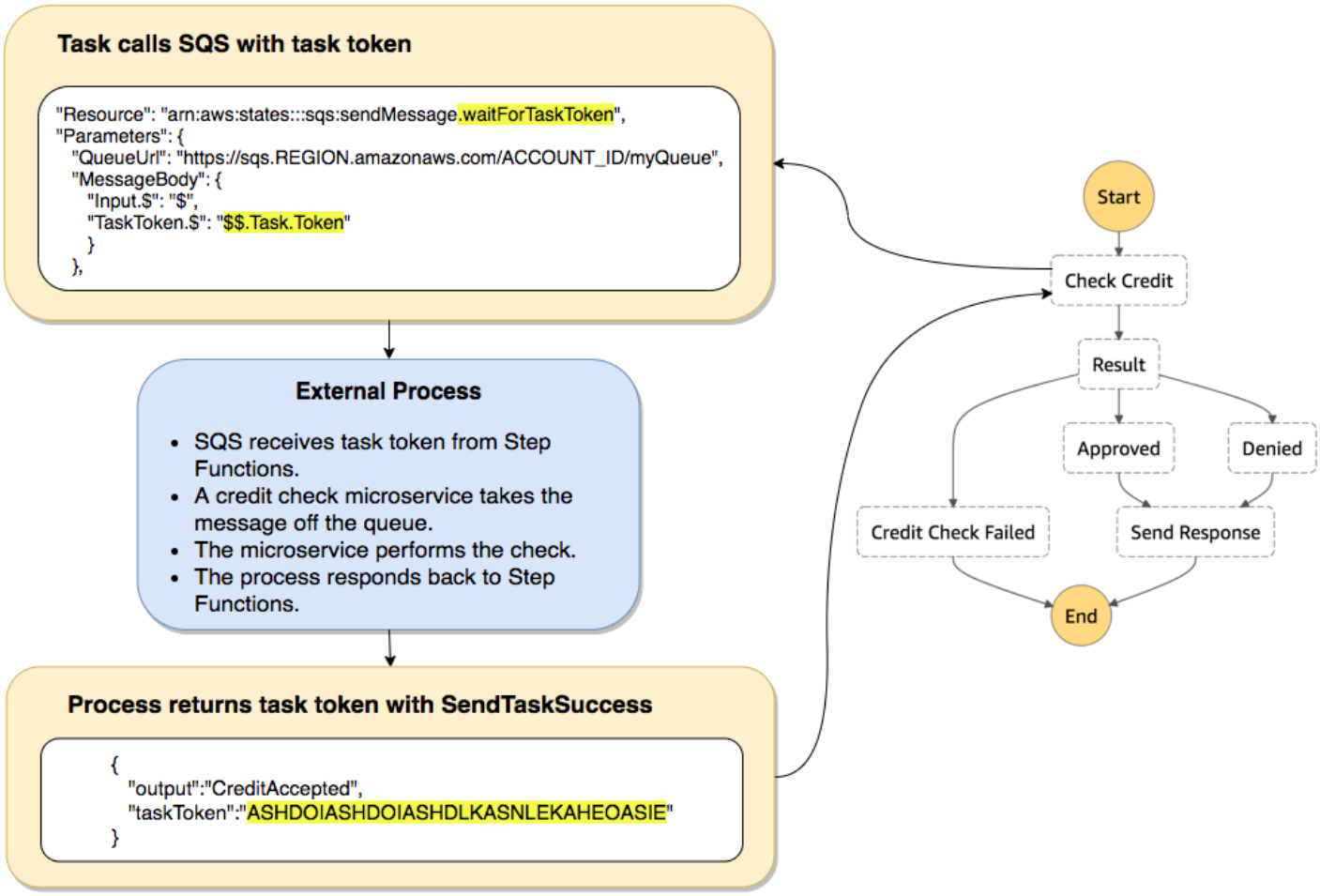
Untuk melihat daftar dukungan layanan terintegrasi yang menunggu token tugas (`.waitForTaskToken`), lihat Integrasi yang dioptimalkan untuk Step Functions.

Topik

- Contoh Token Tugas
- Dapatkan Token dari Object Konteks
- Konfigurasikan Batas Waktu Heartbeat untuk Tugas Menunggu

Contoh Token Tugas

Dalam contoh ini, alur kerja Step Functions perlu diintegrasikan dengan layanan mikro eksternal untuk melakukan pemeriksaan kredit sebagai bagian dari alur kerja persetujuan. Step Functions memublikasikan pesan Amazon SQS yang mencakup token tugas sebagai bagian dari pesan. Sistem eksternal terintegrasi dengan Amazon SQS, dan menarik pesan dari antrean. Ketika selesai, sistem eksternal mengembalikan hasil dan token tugas asli. Step Functions kemudian dilanjutkan dengan alur kerjanya.



Bidang "`Resource`" dari ketentuan tugas yang mereferensikan Amazon SQS termasuk `.waitForTaskToken` yang ditambahkan ke akhir.

```
"Send message to SQS": {
  "Type": "Task",
  "Resource": "arn:aws:states:::sqs:sendMessage.waitForTaskToken",
  "Parameters": {
    "QueueUrl": "https://sqs.us-east-2.amazonaws.com/123456789012/myQueue",
    "MessageBody": {
        "Message": "Hello from Step Functions!",
        "TaskToken.$": "$$.Task.Token"
     }
  },
  "Next": "NEXT_STATE"
}
```

Hal tersebut memberitahu Step Functions untuk menjeda dan menunggu token tugas. Bila Anda menentukan sumber daya menggunakan `.waitForTaskToken`, token tugas dapat diakses di bidang `"Parameters"` ketentuan status Anda dengan penunjukan jalur khusus (`$$.Task.Token`). `$$.` awal menunjukkan bahwa jalur mengakses objek konteks, dan mendapatkan token tugas untuk tugas saat ini dalam eksekusi yang sedang berjalan.

Setelah selesai, layanan eksternal akan memanggil `SendTaskSuccess` atau `SendTaskFailure` termasuk dengan `taskToken`. Baru setelah itu alur kerja berlanjut ke status berikutnya.

> **Note**
>
> Untuk menghindari menunggu tanpa batas waktu jika proses gagal untuk mengirim token tugas dengan `SendTaskSuccess` atau `SendTaskFailure`, lihat Konfigurasikan Batas Waktu Heartbeat untuk Tugas Menunggu.

Dapatkan Token dari Object Konteks

Objek konteks adalah objek JSON internal yang berisi informasi tentang eksekusi Anda. Seperti input status, objek konteks dapat diakses dengan jalur dari bidang `"Parameters"` selama eksekusi. Ketika diakses dari dalam ketentuan tugas, objek konteks mencakup informasi tentang eksekusi tertentu, termasuk token tugas.

```
{
    "Execution": {
        "Id": "arn:aws:states:us-
east-1:123456789012:execution:stateMachineName:executionName",
        "Input": {
           "key": "value"
        },
        "Name": "executionName",
        "RoleArn": "arn:aws:iam::123456789012:role...",
        "StartTime": "2019-03-26T20:14:13.192Z"
    },
    "State": {
        "EnteredTime": "2019-03-26T20:14:13.192Z",
        "Name": "Test",
        "RetryCount": 3
    },
    "StateMachine": {
        "Id": "arn:aws:states:us-east-1:123456789012:stateMachine:stateMachineName",
```

```
        "Name": "name"
    },
    "Task": {
        "Token": "h7XRiCdLtd/83p1E0dMccoxlzFhglsdkzpK9mBVKZsp7d9yrT1W"
    }
}
```

Anda dapat mengakses token tugas dengan menggunakan jalur khusus dari dalam bidang `"Parameters"` dari ketentuan tugas Anda. Untuk mengakses input atau objek konteks, pertama tentukan bahwa parameter akan menjadi jalur dengan menambahkan `.$` ke nama parameter. Berikut ini menentukan simpul dari kedua input dan objek konteks dalam spesifikasi `"Parameters"`.

```
"Parameters": {
   "Input.$": "$",
   "TaskToken.$": "$$.Task.Token"
 },
```

Pada kedua kasus, menambahkan `.$` ke nama parameter memberitahu Step Functions untuk mengharapkan jalur. Pada kasus pertama, `"$"` adalah jalur yang mencakup seluruh input. pada kasus kedua, `$$.` menetapkan bahwa jalur akan mengakses objek konteks, dan `$$.Task.Token` menetapkan parameter untuk nilai token tugas dalam objek konteks eksekusi yang sedang berjalan.

Dalam contoh Amazon SQS, `.waitForTaskToken` di bidang `"Resource"` memberitahu Step Functions untuk menunggu token tugas yang akan dikembalikan. Parameter `"TaskToken.$": "$$.Task.Token"` meneruskan token tersebut sebagai bagian dari pesan Amazon SQS.

```
"Send message to SQS": {
  "Type": "Task",
  "Resource": "arn:aws:states:::sqs:sendMessage.waitForTaskToken",
  "Parameters": {
    "QueueUrl": "https://sqs.us-east-2.amazonaws.com/123456789012/myQueue",
    "MessageBody": {
        "Message": "Hello from Step Functions!",
        "TaskToken.$": "$$.Task.Token"
     }
  },
  "Next": "NEXT_STATE"
}
```

Untuk informasi selengkapnya tentang objek konteks, lihat Objek konteks di bagian Pengolahan Input dan Output dalam panduan ini.

Konfigurasikan Batas Waktu Heartbeat untuk Tugas Menunggu

Tugas yang menunggu token tugas akan menunggu hingga eksekusi mencapai kuota layanan satu tahun (lihat, Kuota terkait throttling status). Untuk menghindari eksekusi yang macet, Anda dapat mengonfigurasi interval batas waktu heartbeat dalam ketentuan mesin status Anda. Gunakan bidang `HeartbeatSeconds` untuk menentukan interval batas waktu.

```
{
  "StartAt": "Push to SQS",
  "States": {
    "Push to SQS": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sqs:sendMessage.waitForTaskToken",
      "HeartbeatSeconds": 600,
      "Parameters": {
        "MessageBody": { "myTaskToken.$": "$$.Task.Token" },
        "QueueUrl": "https://sqs.us-east-1.amazonaws.com/123456789012/push-based-queue"
      },
      "ResultPath": "$.SQS",
      "End": true
    }
  }
}
```

Dalam ketentuan mesin status ini, tugas mendorong pesan ke Amazon SQS dan menunggu proses eksternal untuk memanggil kembali dengan token tugas yang disediakan. Bidang `"HeartbeatSeconds": 600` menetapkan interval batas waktu heartbeat hingga 10 menit. Tugas akan menunggu token tugas yang akan dikembalikan dengan salah satu tindakan API ini:

- `SendTaskSuccess`
- `SendTaskFailure`
- `SendTaskHeartbeat`

Jika tugas menunggu tidak menerima token tugas yang valid dalam periode 10 menit, tugas tersebut akan gagal dengan nama kesalahan `States.Timeout`.

Untuk informasi selengkapnya, lihat Contoh Pola Panggilan Balik (Amazon SQS, Amazon SNS, Lambda) proyek sampel tugas panggilan balik.

Meneruskan parameter ke API layanan

Gunakan bidang `Parameters` di status `Task` untuk mengontrol parameter yang diteruskan ke layanan API.

Di dalam `Parameters` bidang, Anda harus menggunakan bentuk jamak dari parameter array dalam tindakan API. Misalnya, jika Anda menggunakan bidang Filter tindakan `DescribeSnapshots` API untuk mengintegrasikan dengan Amazon EC2, Anda harus menentukan bidang sebagai. `Filters` Jika Anda tidak menggunakan bentuk jamak, Step Functions mengembalikan galat berikut:

```
The field Filter is not supported by Step Functions.
```

Lulus JSON statis sebagai parameter

Anda dapat menyertakan objek JSON langsung dalam ketetapan mesin status Anda untuk melewatkan sebagai parameter ke sumber daya.

Misalnya, untuk mengatur parameter `RetryStrategy` untuk API `SubmitJob` untuk AWS Batch, Anda dapat menyertakan hal-hal berikut dalam parameter Anda.

```
"RetryStrategy": {
  "attempts": 5
}
```

Anda juga dapat meneruskan beberapa parameter dengan JSON statis. Sebagai contoh yang lebih lengkap, berikut ini adalah `Resource` dan `Parameters` bidang spesifikasi tugas yang menerbitkan ke topik Amazon SNS bernama. *myTopic*

```
"Resource": "arn:aws:states:::sns:publish",
  "Parameters": {
     "TopicArn": "arn:aws:sns:us-east-2:123456789012:myTopic",
     "Message": "test message",
     "MessageAttributes": {
       "my attribute no 1": {
         "DataType": "String",
         "StringValue": "value of my attribute no 1"
       },
       "my attribute no 2": {
         "DataType": "String",
         "StringValue": "value of my attribute no 2"
       }
```

```
    }
  },
```

Lulus masukan status sebagai parameter menggunakan Jalur

Anda dapat melewati bagian dari masukan negara sebagai parameter dengan menggunakan jalur. Jalur adalah string, diawali dengan $, yang digunakan untuk mengidentifikasi komponen dalam teks JSON. Langkah Fungsi jalan menggunakan JsonPathsintaks.

Untuk menentukan bahwa parameter menggunakan jalur, akhiri nama parameter dengan `.$`. Misalnya, jika input state Anda berisi teks dalam node bernama`message`, Anda bisa meneruskan teks itu sebagai parameter menggunakan path.

Pertimbangkan masukan negara berikut:

```
{
  "comment": "A message in the state input",
  "input": {
    "message": "foo",
    "otherInfo": "bar"
  },
  "data": "example"
}
```

Untuk meneruskan nilai node bernama `message` sebagai parameter, tentukan sintaks berikut:

```
"Parameters": {"myMessage.$": "$.input.message"},
```

Langkah Fungsi kemudian melewati nilai `foo` sebagai parameter.

Untuk informasi selengkapnya tentang cara menggunakan parameter dalam Step Functions, lihat hal-hal berikut ini:

- Pengolahan Input dan Output
- InputPath, Parameter dan ResultSelector

Meneruskan Simpul Objek Konteks sebagai Parameter

Selain konten statis, dan simpul dari input status, Anda dapat meneruskan simpul dari objek konteks sebagai parameter. Objek konteks adalah data JSON dinamis yang ada selama eksekusi mesin

status. Ini mencakup informasi tentang mesin status Anda dan eksekusi saat ini. Anda dapat mengakses objek konteks menggunakan jalur di kolom `"Parameters"` ketetapan status.

Untuk informasi selengkapnya tentang objek konteks dan cara mengakses data tersebut kolom `"Parameters"`, lihat hal-hal berikut ini:

- Objek konteks
- Mengakses Obyek Konteks
- Dapatkan Token dari Object Konteks

Ubah log untuk integrasi AWS SDK yang didukung

Tabel berikut merangkum kapan layanan awalnya terintegrasi dengan Step Functions dan kapan API integrasinya terbaru diperbarui. Untuk detail tentang penggunaan integrasi, lihatAWS Integrasi layanan SDK.

> **Important**
>
> Dukungan tindakan API dirilis pada irama triwulanan. Pembaruan untuk tindakan yang sudah didukung, seperti parameter baru, mungkin tidak segera tersedia.

Layanan	Dukungan awal	Diperbarui	
AWS AppFabric	Januari 18, 2024		
B2B Data Interchange	Januari 18, 2024		
Ekspor Data AWS	Januari 18, 2024		
Amazon Bedrock	Januari 18, 2024		
Amazon Bedrock Agents	Januari 18, 2024		
Amazon Bedrock Runtime Agents	Januari 18, 2024		

Layanan	Dukungan awal	Diperbarui	
Amazon Bedrock Runtime	Januari 18, 2024		
Amazon CloudFront KeyValueStore	Januari 18, 2024		
Amazon CodeGuru Security	Januari 18, 2024		
Hub Optimisasi Biaya AWS	Januari 18, 2024		
Amazon DataZone	Januari 18, 2024		
Amazon EKS Auth	Januari 18, 2024		
Resolusi Entitas AWS	Januari 18, 2024		
AWS Tingkat Gratis	Januari 18, 2024		
Amazon Inspector Scan	Januari 18, 2024		
AWS Launch Wizard	Januari 18, 2024		
Amazon Managed Blockchain Query	Januari 18, 2024		
AWS Elemental MediaPackage V2	Januari 18, 2024		
AWS HealthImaging	Januari 18, 2024		
Network Manager	Januari 18, 2024		
AWS Payment Cryptography	Januari 18, 2024		

Layanan	Dukungan awal	Diperbarui	
AWS Payment Cryptography Data	Januari 18, 2024		
AWS Private CA Connector for Active Directory	Januari 18, 2024		
Amazon Q Business	Januari 18, 2024		
Amazon Q Connect	Januari 18, 2024		
AWS re:Post	Januari 18, 2024		
Amazon Timestream Query	Januari 18, 2024		
Amazon Timestream Write	Januari 18, 2024		
Trusted Advisor	Januari 18, 2024		
Verified Permissions	Januari 18, 2024		
Amazon WorkSpaces Thin Client	Januari 18, 2024		
AWS CloudTrail Data	Juni 16, 2023		
Amazon CloudWatch Internet Monitor	Juni 16, 2023		
Amazon Interacti ve Video Service RealTime	Juni 16, 2023		
AWS IoT TwinMaker	Juni 16, 2023		

Layanan	Dukungan awal	Diperbarui	
Amazon OpenSearch Ingestion	Juni 16, 2023		
AWS Telco Network Builder	Juni 16, 2023		
Amazon VPC Lattice	Juni 16, 2023		
AWS Backup Storage	17 Februari 2023		
Amazon Chime Media Pipelines	17 Februari 2023		
Amazon Chime Voice	17 Februari 2023		
AWS Clean Rooms	17 Februari 2023	Januari 18, 2024	
Amazon CodeCatalyst	17 Februari 2023		
Amazon Connect Cases	17 Februari 2023		
AWS Control Tower	17 Februari 2023		
Amazon DocumentDB Elastic Clusters	17 Februari 2023		
Amazon EMR Serverless	17 Februari 2023		
Amazon IVS Chat	17 Februari 2023		
Amazon Kendra Intelligent Ranking	17 Februari 2023		
AWS HealthOmics	17 Februari 2023		
Amazon Redshift Serverless	17 Februari 2023		

Layanan	Dukungan awal	Diperbarui	
Amazon Security Lake	17 Februari 2023		
AWS Health	17 Februari 2023		
AWS IoT FleetWise	17 Februari 2023		
AWS IoT RoboRunner	17 Februari 2023		
AWS Mainframe Modernization	17 Februari 2023		
Orkestrator AWS Migration Hub	17 Februari 2023		
AWS Private 5G	17 Februari 2023		
Penjelajah Sumber Daya AWS	17 Februari 2023		
AWS SimSpace Weaver	17 Februari 2023		
AWS Support App	17 Februari 2023		
CloudWatch Observability Access Manager	17 Februari 2023		
EventBridge Pipes	17 Februari 2023		
EventBridge Scheduler	17 Februari 2023		
IAM Roles Anywhere	17 Februari 2023		
Kinesis Video WebRTC Storage	17 Februari 2023		

Layanan	Dukungan awal	Diperbarui	
License Manager Linux Subscriptions	17 Februari 2023		
License Manager User Subscriptions	17 Februari 2023		
OpenSearch Serverless	17 Februari 2023		
Route 53 ARC Zonal Shift	17 Februari 2023		
SageMaker Geospatial	17 Februari 2023		
SageMaker Metrics	17 Februari 2023		
Systems Manager for SAP	17 Februari 2023		
AWS Account Management	19 April 2022		
AWS Amplify	30 September 2021		
AWS App Mesh	30 September 2021		
AWS App Runner	30 September 2021	17 Februari 2023	
AWS AppConfig	30 September 2021		
AWS AppConfig Data	19 April 2022		
AWS AppSync	30 September 2021	17 Februari 2023	
AWS Application Discovery Service	30 September 2021		

Layanan	Dukungan awal	Diperbarui	
AWS Application Migration Service	30 September 2021		
AWS Audit Manager	30 September 2021		
AWS Auto Scaling Plans	30 September 2021		
AWS Backup	30 September 2021	17 Februari 2023	
AWS Backup gateway	19 April 2022	17 Februari 2023	
AWS Batch	30 September 2021	17 Februari 2023	
AWS Billing Conductor	26 Juli 2022	17 Februari 2023	
AWS Budgets	30 September 2021		
AWS Certificate Manager	30 September 2021		
AWS Private Certifica te Authority	30 September 2021		
AWS Cloud Map	30 September 2021		
AWS Cloud9	30 September 2021		
AWS CloudFormation	30 September 2021	17 Februari 2023	
AWS CloudHSM	30 September 2021		
AWS CloudHSM	30 September 2021		
AWS CloudTrail	30 September 2021	17 Februari 2023	
AWS Cloud Control	19 April 2022		

Layanan	Dukungan awal	Diperbarui	
AWS CodeBuild	30 September 2021		
AWS CodeCommit	30 September 2021	17 Februari 2023	
AWS CodeDeploy	30 September 2021		
AWS CodePipeline	30 September 2021		
AWS CodeStar	30 September 2021		
AWS CodeStar	30 September 2021		
AWS CodeStar	30 September 2021		
AWS Compute Optimizer	30 September 2021	17 Februari 2023	
AWS Config	30 September 2021	26 Juli 2022	
AWS Cost Explorer Service	30 September 2021	17 Februari 2023	
AWS Cost and Usage Report	30 September 2021		
AWS Data Exchange	30 September 2021	26 Juli 2022	
AWS Data Pipeline	30 September 2021		
AWS DataSync	30 September 2021	26 Juli 2022	
AWS Database Migration Service	30 September 2021		
AWS Device Farm	30 September 2021		
AWS Direct Connect	30 September 2021		

Layanan	Dukungan awal	Diperbarui	
AWS Directory Service	30 September 2021	17 Februari 2023	
AWS EC2 Instance Connect	30 September 2021		
AWS Elastic Beanstalk	30 September 2021		
AWS Elemental MediaLive	30 September 2021		
AWS Elemental MediaPackage	30 September 2021		
AWS Elemental MediaPackage VOD	30 September 2021		
AWS Elemental MediaStore	30 September 2021		
AWS Fault Injection Service	30 September 2021		
AWS Firewall Manager	30 September 2021	17 Februari 2023	
AWS Glue	30 September 2021	17 Februari 2023	
AWS Glue DataBrew	30 September 2021		
AWS IoT Greengrass	30 September 2021		
AWS Ground Station	30 September 2021	17 Februari 2023	
AWS Identity and Access Management	30 September 2021		

Layanan	Dukungan awal	Diperbarui	
AWS IoT	30 September 2021	17 Februari 2023	
AWS IoT 1-Click	30 September 2021		
AWS IoT Analytics	30 September 2021		
AWS IoT Core Device Advisor	30 September 2021		
AWS IoT Events	30 September 2021		
Data AWS IoT Events	30 September 2021		
AWS IoT Fleet Hub	30 September 2021		
AWS IoT Greengrass Version 2	30 September 2021		
AWS IoT Jobs Data Plane	30 September 2021		
AWS IoT Secure Tunneling	30 September 2021		
AWS IoT SiteWise	30 September 2021	17 Februari 2023	
AWS IoT Things Graph	30 September 2021		
AWS IoT Wireless	30 September 2021	17 Februari 2023	
AWS Key Management Service	30 September 2021	26 Juli 2022	
AWS Lake Formation	30 September 2021	17 Februari 2023	
AWS Lambda	30 September 2021	17 Februari 2023	

Layanan	Dukungan awal	Diperbarui	
AWS License Manager	30 September 2021	17 Februari 2023	
AWS Marketplace	30 September 2021	17 Februari 2023	
AWS Marketplace Commerce Analytics	30 September 2021		
AWS Marketplace Entitlement Service	30 September 2021		
AWS Elemental MediaTailor	30 September 2021	26 Juli 2022	
AWS Migration Hub	30 September 2021		
AWS Migration Hub Config	30 September 2021		
Rekomendasi Strategi AWS Migration Hub	19 April 2022	17 Februari 2023	
AWS Mobile	30 September 2021		
AWS Network Firewall	30 September 2021		
AWS OpsWorks	30 September 2021		
AWS OpsWorks CM	30 September 2021		
AWS Organizations	30 September 2021	17 Februari 2023	
AWS Outposts	30 September 2021		
AWS Panorama	19 April 2022	17 Februari 2023	
Amazon Relationa l Database Service Performance Insights	30 September 2021		

Layanan	Dukungan awal	Diperbarui	
Daftar Harga AWS	30 September 2021		
Amazon Relational Database Service	30 September 2021		
AWS Resilience Hub	19 April 2022		
AWS Resource Access Manager	30 September 2021		
AWS Resource Groups	30 September 2021	17 Februari 2023	
AWS Resource Groups Tagging API	30 September 2021		
AWS RoboMaker	30 September 2021		
AWS IAM Identity Center	30 September 2021	17 Februari 2023	
AWS SSO OIDC	30 September 2021		
AWS Secrets Manager	30 September 2021		
AWS Security Token Service	30 September 2021		
AWS Security Hub	30 September 2021		
AWS Server Migration Service	30 September 2021		
AWS Service Catalog	30 September 2021		
AWS Service Catalog AppRegistry	30 September 2021	17 Februari 2023	

Layanan	Dukungan awal	Diperbarui	
AWS Shield	30 September 2021		
AWS Signer	30 September 2021		
AWS IAM Identity Center	30 September 2021		
AWS IAM Identity Center Admin	30 September 2021		
AWS Step Functions	30 September 2021	17 Februari 2023	
AWS Storage Gateway	30 September 2021		
AWS Support	30 September 2021		
AWS Transfer Family	30 September 2021	17 Februari 2023	
AWS WAF	30 September 2021		
AWS WAF Regional	30 September 2021		
AWS WAFV2	30 September 2021		
AWS Well-Architected Tool	30 September 2021	17 Februari 2023	
AWS X-Ray	30 September 2021	17 Februari 2023	
AWS Marketplace Metering Service	30 September 2021		
AWS Serverless Application Repository	30 September 2021		
AWS Identity and Access Management Access Analyzer	30 September 2021		

Layanan	Dukungan awal	Diperbarui	
Alexa for Business	30 September 2021		
Amazon API Gateway	30 September 2021	17 Februari 2023	
Amazon API Gateway	30 September 2021		
Amazon AppIntegrations	30 September 2021		
Amazon AppStream 2.0	30 September 2021		
Amazon AppFlow	30 September 2021	17 Februari 2023	
Amazon Athena	30 September 2021	17 Februari 2023	
Amazon Augmented AI	30 September 2021		
Amazon Braket	30 September 2021		
Amazon Chime	30 September 2021		
Amazon Chime Meetings	19 April 2022	17 Februari 2023	
Amazon Cloud Directory	30 September 2021		
Amazon CloudFront	30 September 2021	17 Februari 2023	
Amazon CloudSearch	30 September 2021		
Amazon CloudWatch	30 September 2021	17 Februari 2023	
Amazon CloudWatch Application Insights	30 September 2021		
CloudWatch Evidently	19 April 2022		

Layanan	Dukungan awal	Diperbarui	
Amazon CloudWatch Logs	30 September 2021		
Amazon CloudWatch RUM	19 April 2022	17 Februari 2023	
Amazon CloudWatch Synthetics	30 September 2021		
Amazon CodeGuru Profiler	30 September 2021		
Amazon CodeGuru Reviewer	30 September 2021		
Amazon Cognito	30 September 2021		
Amazon Cognito Identity Provider	30 September 2021		
Amazon Cognito Sync	30 September 2021		
Amazon Comprehend	30 September 2021	17 Februari 2023	
Amazon Comprehend Medical	30 September 2021		
Amazon Connect Contact Lens	30 September 2021		
Amazon Connect Participant Service	30 September 2021		
Amazon Connect	30 September 2021	17 Februari 2023	
Amazon Connect Voice ID	19 April 2022		

Layanan	Dukungan awal	Diperbarui	
Amazon Connect Wisdom	19 April 2022		
Amazon Data Lifecycle Manager	30 September 2021		
Amazon Detective	30 September 2021		
Amazon DevOps Guru	30 September 2021	26 Juli 2022	
Amazon DocumentDB (with MongoDB compatibility)	30 September 2021		
Amazon DynamoDB	30 September 2021	17 Februari 2023	
Amazon DynamoDB Streams	30 September 2021		
Amazon EC2 Container Registry	30 September 2021		
Amazon EC2 Container Service	30 September 2021	17 Februari 2023	
Amazon EC2 Systems Manager	30 September 2021	17 Februari 2023	
Amazon EMR	30 September 2021	17 Februari 2023	
Amazon ElastiCache	30 September 2021		
Amazon Elastic Inference	30 September 2021		
Amazon Elastic Block Store	30 September 2021		

Layanan	Dukungan awal	Diperbarui	
Amazon Elastic Compute Cloud	30 September 2021	17 Februari 2023	
Amazon Elastic Container Registry Public	30 September 2021		
Amazon Elastic File System	30 September 2021		
Amazon Elastic Kubernetes Service	30 September 2021	17 Februari 2023	
Amazon EMR	30 September 2021		
Amazon Elastic Transcoder	30 September 2021		
Amazon OpenSearch Service	30 September 2021	17 Februari 2023	
Amazon OpenSearch Service	19 April 2022	17 Februari 2023	
Amazon EventBridge	30 September 2021	17 Februari 2023	
Amazon FSx	30 September 2021	17 Februari 2023	
Amazon Forecast Query	30 September 2021	17 Februari 2023	
Amazon Forecast Service	30 September 2021	17 Februari 2023	
Amazon Fraud Detector	30 September 2021		
Amazon GameLift	30 September 2021	17 Februari 2023	

Layanan	Dukungan awal	Diperbarui	
Amazon GameSparks	26 Juli 2022		
Amazon S3 Glacier	30 September 2021		
Amazon GuardDuty	30 September 2021		
AWS HealthLake	30 September 2021		
Amazon Honeycode	30 September 2021		
Amazon Inspector	30 September 2021		
Amazon Inspector V2	19 April 2022		
Amazon Interactive Video Service	30 September 2021		
Amazon Kendra	30 September 2021		
Amazon Kinesis	30 September 2021		
Amazon Kinesis Analytics	30 September 2021		
Amazon Kinesis Analytics V2	30 September 2021		
Amazon Kinesis Firehose	30 September 2021		
Amazon Kinesis Video Signaling Channels	30 September 2021		
Amazon Kinesis Video Streams	30 September 2021	17 Februari 2023	

Layanan	Dukungan awal	Diperbarui	
Amazon Kinesis Video Streams Archived Media	30 September 2021		
Amazon Kinesis video stream	30 September 2021		
Amazon Lex Model Building Service	30 September 2021		
Amazon Lex Model Building Service V2	30 September 2021	17 Februari 2023	
Amazon Lex	30 September 2021		
Amazon Lex Runtime V2	30 September 2021		
Amazon Lightsail	30 September 2021	17 Februari 2023	
Amazon Location Service	30 September 2021	17 Februari 2023	
Amazon Lookout for Equipment	30 September 2021		
Amazon Lookout for Metrics	30 September 2021	17 Februari 2023	
Amazon Lookout for Vision	30 September 2021		
Amazon MQ	30 September 2021		
Amazon Macie	30 September 2021		
Amazon Macie 2	30 September 2021	17 Februari 2023	

Layanan	Dukungan awal	Diperbarui	
Amazon Managed Blockchain	30 September 2021	17 Februari 2023	
Amazon Managed Grafana	19 April 2022	17 Februari 2023	
Amazon Managed Service for Prometheus	30 September 2021	17 Februari 2023	
Amazon Managed Streaming for Apache Kafka	30 September 2021	17 Februari 2023	
Amazon Managed Streaming for Apache Kafkax	19 April 2022		
Amazon Managed Workflows for Apache Airflow	30 September 2021		
Amazon Mechanical Turk	30 September 2021		
Amazon MemoryDB for Redis	19 April 2022	17 Februari 2023	
Amazon Nimble Studio	30 September 2021		
Amazon Personalize	30 September 2021	17 Februari 2023	
Amazon Personalize Events	30 September 2021		

Layanan	Dukungan awal	Diperbarui	
Amazon Personalize Runtime	30 September 2021		
Amazon Pinpoint	30 September 2021		
Amazon Pinpoint Email Service	30 September 2021		
Amazon Pinpoint SMS and Voice Service	30 September 2021		
Amazon Pinpoint SMS and Voice V2 Service	26 Juli 2022		
Amazon Polly	30 September 2021		
Amazon QLDB	30 September 2021		
Amazon QLDB Session	30 September 2021		
Amazon QuickSight	30 September 2021	17 Februari 2023	
Amazon Redshift	30 September 2021		
Amazon Redshift Data API	30 September 2021		
Amazon Rekognition	30 September 2021	17 Februari 2023	
Amazon Relational Database Service	30 September 2021	17 Februari 2023	
Amazon Route 53	30 September 2021		

Layanan	Dukungan awal	Diperbarui	
Amazon Route 53 Recovery Control Config	30 September 2021	26 Juli 2022	
Amazon Route 53 Domains	30 September 2021	17 Februari 2023	
Amazon Route 53 Resolver	30 September 2021		
Amazon S3 on Outposts	30 September 2021	26 Juli 2022	
Amazon SageMaker Runtime Feature Store Runtime	30 September 2021		
Amazon SageMaker Runtime Runtime	30 September 2021		
Amazon SageMaker	30 September 2021	17 Februari 2023	
Amazon SageMaker Edge Manager	30 September 2021		
Amazon Simple Email Service	30 September 2021		
Amazon Simple Email Service V2	30 September 2021	17 Februari 2023	
Amazon Simple Notification Service	30 September 2021	17 Februari 2023	
Amazon Simple Queue Service	30 September 2021	17 Februari 2023	

Layanan	Dukungan awal	Diperbarui	
Amazon Simple Storage Service	30 September 2021	17 Februari 2023	
Amazon Simple Workflow Service	30 September 2021		
Amazon Textract	30 September 2021	17 Februari 2023	
Amazon Transcribe	30 September 2021		
Amazon Translate	30 September 2021	17 Februari 2023	
Amazon WorkDocs	30 September 2021	17 Februari 2023	
Amazon WorkMail	30 September 2021	17 Februari 2023	
Amazon WorkMail Message Flow	30 September 2021		
Amazon WorkSpaces	30 September 2021	17 Februari 2023	
Amazon WorkSpaces Web	19 April 2022	17 Februari 2023	
Amplify	30 September 2021		
Amplify UI Builder	19 April 2022	17 Februari 2023	
Application Auto Scaling	30 September 2021		
Amazon EC2 Auto Scaling	30 September 2021	17 Februari 2023	
CodeArtifact	30 September 2021		
DynamoDB Accelerator	30 September 2021		

Layanan	Dukungan awal	Diperbarui	
EC2 Image Builder	30 September 2021		
AWS Elastic Disaster Recovery	19 April 2022	17 Februari 2023	
Elastic Load Balancing	30 September 2021		
Elastic Load Balancing V2	30 September 2021		
MediaConnect	30 September 2021		
Amazon S3 Control	30 September 2021	17 Februari 2023	
Recycle Bin for Amazon EBS	19 April 2022	17 Februari 2023	
Savings Plans	30 September 2021		
Amazon EventBridge Schema Registry	30 September 2021		
Service Quotas	30 September 2021		
AWS Snowball	30 September 2021		

Proyek sampel untuk Step Functions

Di AWS Step Functionskonsol, Anda dapat memilih salah satu template pemula berikut untuk menyebarkan mesin status ke AndaAkun AWS. Template pemula ini adalah ready-to-run contoh proyek yang secara otomatis membuat proptotype dan definisi alur kerja, dan semua AWS sumber daya terkait untuk proyek tersebut.

Anda dapat menggunakan proyek contoh ini untuk menyebarkan dan menjalankannya apa adanya, atau menggunakan prototipe alur kerja untuk membangunnya. Jika Anda membangun proyek ini, Step Functions membuat prototipe alur kerja, tetapi tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

Saat Anda menerapkan proyek sampel, mereka menyediakan mesin status yang berfungsi penuh, dan membuat sumber daya terkait untuk menjalankan mesin status. Ketika Anda membuat proyek sampel, Step Functions menggunakan AWS CloudFormation untuk membuat sumber daya terkait direferensikan oleh mesin status.

Topik

- Mengelola pekerjaan batch (AWS Batch,Amazon SNS)
- Mengelola tugas kontainer (Amazon ECS,Amazon SNS)
- Mentransfer catatan data (Lambda,DynamoDB,Amazon SQS)
- Polling untuk Status Pekerjaan (Lambda,) AWS Batch
- Timer Tugas (Lambda, Amazon SNS)
- Contoh Pola Panggilan Balik (Amazon SQS, Amazon SNS, Lambda)
- Kelola Tugas Amazon EMR
- Jalankan EMR Serverless pekerjaan
- Mulai Alur Kerja dalam Alur Kerja (Step Functions, Lambda)
- Secara dinamis memproses data dengan status Peta
- Memproses file CSV dengan Peta Terdistribusi
- Memproses data dalam bucket Amazon S3 dengan Peta Terdistribusi
- Melatih Model Machine Learning
- Setel Model Machine Learning
- Proses Pesan Bervolume Tinggi dari Amazon SQS (Alur Kerja Express)
- Contoh Checkpointing selektif (Alur kerja Express)

- Membangun AWS CodeBuild Proyek (CodeBuild, Amazon SNS)
- Praproses data dan latih model machine learning
- Contoh orkestrasi Lambda
- Mulai kueri Athena
- Jalankan beberapa kueri (Amazon Athena, Amazon SNS)
- Kueri kumpulan data besar (Amazon Athena, Amazon S3,, AWS Glue Amazon SNS)
- Tetap perbarui data (Amazon Athena, Amazon S3,) AWS Glue
- Mengelola klaster Amazon EKS
- Buat panggilan ke API Gateway
- Panggil layanan mikro yang berjalan di Fargate menggunakan integrasi API Gateway
- Kirim acara khusus ke EventBridge
- Memanggil Alur Kerja Express Sinkron
- Jalankan alur kerja ETL/ELT menggunakan Amazon Redshift (Lambda, API Data Amazon Redshift)
- Gunakan Step Functions dan AWS Batch dengan penanganan kesalahan
- Menggemari AWS Batch pekerjaan
- AWS Batch dengan Lambda
- Lakukan AI prompt-chaining dengan Amazon Bedrock

Mengelola pekerjaan batch (AWS Batch,Amazon SNS)

Contoh proyek ini menunjukkan cara mengirimkan AWS Batch pekerjaan, dan kemudian mengirim Amazon SNS pemberitahuan berdasarkan apakah pekerjaan itu berhasil atau gagal. Menyebarkan proyek sampel ini akan menciptakan mesin AWS Step Functions status, AWS Batch pekerjaan, dan Amazon SNS topik.

Dalam proyek ini, Step Functions menggunakan mesin negara untuk memanggil AWS Batch pekerjaan secara serempak. Kemudian menunggu pekerjaan berhasil atau gagal, dan mengirimkan Amazon SNS topik dengan pesan tentang apakah pekerjaan itu berhasil atau gagal.

Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan

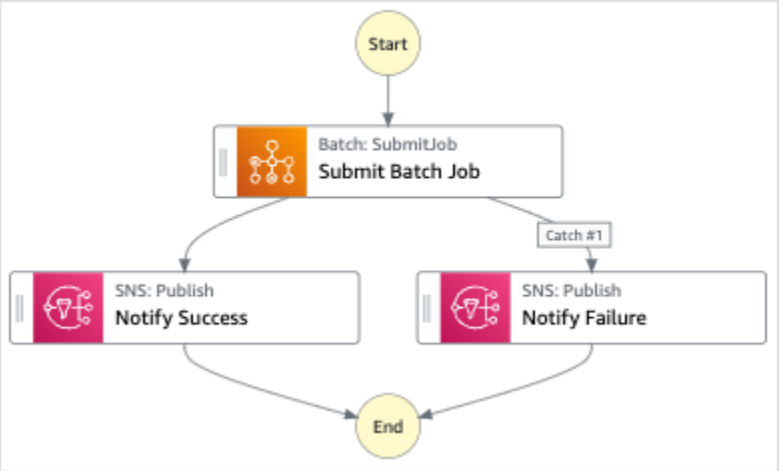
1. Buka Konsol Step Functions dan pilih Buat mesin status.

2. Ketik **`Manage a batch job`** di kotak pencarian, lalu pilih Kelola pekerjaan batch dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

   Proyek contoh ini menyebarkan sumber daya berikut:

   - Sebuah AWS Batch pekerjaan
   - Topik Amazon SNS
   - Mesin AWS Step Functions negara
   - Peran terkait AWS Identity and Access Management (IAM)

   Gambar berikut menunjukkan grafik alur kerja untuk Mengelola proyek sampel pekerjaan batch:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

     Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon

States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

> **Important**
>
> Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

  > **Tip**
  >
  > Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

  Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

  Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

  Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

  > **Important**
  >
  > Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

1. Pada halaman mesin Negara, pilih proyek sampel Anda.

2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

    1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

        > **Note**
        >
        > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

    2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

        Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

        > **Note**
        >
        > Jika proyek demo yang Anda gunakan berisi data input eksekusi yang telah diisi sebelumnya, gunakan input tersebut untuk menjalankan mesin status.

    3. Pilih Mulai Eksekusi.
    4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

        Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Contoh Kode Mesin Status

Mesin status dalam proyek sampel ini terintegrasi dengan AWS Batch dan Amazon SNS dengan meneruskan parameter langsung ke sumber daya tersebut.

Jelajahi contoh mesin status ini untuk melihat bagaimana Step Functions mengontrol AWS Batch dan Amazon SNS dengan menghubungkan ke Amazon Resource Name (ARN) di `Resource` bidang, dan dengan meneruskan `Parameters` ke API layanan.

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

```
{
  "Comment": "An example of the Amazon States Language for notification on an AWS Batch
 job completion",
  "StartAt": "Submit Batch Job",
  "TimeoutSeconds": 3600,
  "States": {
    "Submit Batch Job": {
      "Type": "Task",
      "Resource": "arn:aws:states:::batch:submitJob.sync",
      "Parameters": {
        "JobName": "BatchJobNotification",
        "JobQueue": "arn:aws:batch:us-east-1:123456789012:job-queue/
BatchJobQueue-7049d367474b4dd",
        "JobDefinition": "arn:aws:batch:us-east-1:123456789012:job-definition/
BatchJobDefinition-74d55ec34c4643c:1"
      },
      "Next": "Notify Success",
      "Catch": [
          {
            "ErrorEquals": [ "States.ALL" ],
            "Next": "Notify Failure"
          }
      ]
    },
    "Notify Success": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
        "Message": "Batch job submitted through Step Functions succeeded",
        "TopicArn": "arn:aws:sns:us-east-1:123456789012:batchjobnotificatiointemplate-
SNSTopic-1J757CVBQ2KHM"
```

```
      },
      "End": true
    },
    "Notify Failure": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
        "Message": "Batch job submitted through Step Functions failed",
        "TopicArn": "arn:aws:sns:us-east-1:123456789012:batchjobnotificatiointemplate-
SNSTopic-1J757CVBQ2KHM"
      },
      "End": true
    }
  }
}
```

Contoh IAM

Contoh kebijakan AWS Identity and Access Management (IAM) yang dihasilkan oleh proyek sampel ini mencakup hak istimewa paling sedikit yang diperlukan untuk mengeksekusi mesin negara dan sumber daya terkait. Kami merekomendasikan agar Anda hanya menyertakan izin yang diperlukan dalam kebijakan IAM Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:ap-northeast-1:123456789012:ManageBatchJob-SNSTopic-
JHLYYG7AZPZI"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "batch:SubmitJob",
                "batch:DescribeJobs",
                "batch:TerminateJob"
            ],
            "Resource": "*",
```

```
            "Effect": "Allow"
        },
        {
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:ap-northeast-1:123456789012:rule/
StepFunctionsGetEventsForBatchJobsRule"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Mengelola tugas kontainer (Amazon ECS,Amazon SNS)

Contoh proyek ini menunjukkan cara menjalankan AWS Fargate tugas, dan kemudian mengirim Amazon SNS pemberitahuan berdasarkan apakah pekerjaan itu berhasil atau gagal. Menyebarkan proyek sampel ini akan membuat mesin AWS Step Functions status, Fargate cluster, dan Amazon SNS topik.

Dalam proyek ini, Step Functions menggunakan mesin negara untuk memanggil Fargate tugas secara sinkron. Kemudian menunggu tugas berhasil atau gagal, dan mengirimkan Amazon SNS topik dengan pesan tentang apakah pekerjaan itu berhasil atau gagal.

Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan

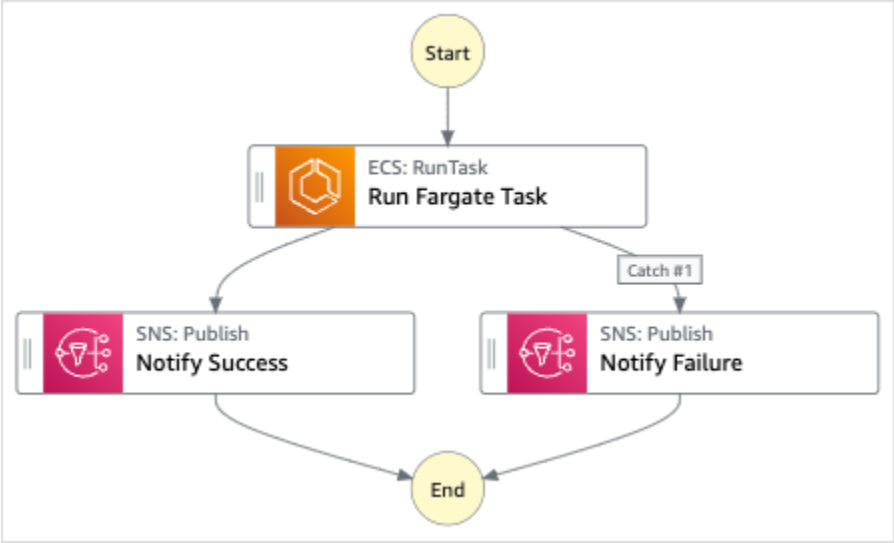
1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`Manage a container task`** di kotak pencarian, lalu pilih Kelola tugas penampung dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke

Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

Proyek contoh ini menyebarkan sumber daya berikut:

- Sebuah AWS Fargate cluster
- Topik Amazon SNS
- Mesin AWS Step Functions negara
- Peran terkait AWS Identity and Access Management (IAM)

Gambar berikut menunjukkan grafik alur kerja untuk Mengelola proyek sampel tugas kontainer:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

- Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

  Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

> **Important**
>
> Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

> **Tip**
>
> Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

> **Important**
>
> Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

1. Pada halaman mesin Negara, pilih proyek sampel Anda.
2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

   > **Note**
   >
   > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

   Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

   > **Note**
   >
   > Jika proyek demo yang Anda gunakan berisi data input eksekusi yang telah diisi sebelumnya, gunakan input tersebut untuk menjalankan mesin status.

3. Pilih Mulai Eksekusi.
4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

   Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Contoh Kode Mesin Status

Mesin status dalam proyek sampel ini terintegrasi dengan AWS Fargate dan Amazon SNS dengan meneruskan parameter langsung ke sumber daya tersebut. Telusuri melalui mesin status contoh ini untuk melihat cara Step Functions menggunakan mesin status untuk memanggil tugas Fargate

secara serentak, menunggu tugas untuk berhasil atau gagal, dan mengirimkan topik Amazon SNS dengan pesan tentang apakah tugas berhasil atau gagal.

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

```
{
  "Comment": "An example of the Amazon States Language for notification on an AWS
 Fargate task completion",
  "StartAt": "Run Fargate Task",
  "TimeoutSeconds": 3600,
  "States": {
    "Run Fargate Task": {
      "Type": "Task",
      "Resource": "arn:aws:states:::ecs:runTask.sync",
      "Parameters": {
        "LaunchType": "FARGATE",
        "Cluster": "arn:aws:ecs:ap-northeast-1:123456789012:cluster/
FargateTaskNotification-ECSCluster-VHLR20IF9IMP",
        "TaskDefinition": "arn:aws:ecs:ap-northeast-1:123456789012:task-definition/
FargateTaskNotification-ECSTaskDefinition-13YOJT8Z2LY5Q:1",
        "NetworkConfiguration": {
          "AwsvpcConfiguration": {
            "Subnets": [
              "subnet-07e1ad3abcfce6758",
              "subnet-04782e7f34ae3efdb"
            ],
            "AssignPublicIp": "ENABLED"
          }
        }
      },
      "Next": "Notify Success",
      "Catch": [
          {
            "ErrorEquals": [ "States.ALL" ],
            "Next": "Notify Failure"
          }
      ]
    },
    "Notify Success": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
```

```
        "Message": "AWS Fargate Task started by Step Functions succeeded",
        "TopicArn": "arn:aws:sns:ap-northeast-1:123456789012:FargateTaskNotification-
SNSTopic-1XYW5YD5V0M7C"
      },
      "End": true
    },
    "Notify Failure": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
        "Message": "AWS Fargate Task started by Step Functions failed",
        "TopicArn": "arn:aws:sns:ap-northeast-1:123456789012:FargateTaskNotification-
SNSTopic-1XYW5YD5V0M7C"
      },
      "End": true
    }
  }
}
```

Contoh IAM

Contoh kebijakan AWS Identity and Access Management (IAM) yang dihasilkan oleh proyek sampel ini mencakup hak istimewa paling sedikit yang diperlukan untuk mengeksekusi mesin negara dan sumber daya terkait. Ini adalah praktik terbaik untuk menyertakan hanya izin yang diperlukan dalam kebijakan IAM Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:ap-northeast-1:123456789012:FargateTaskNotification-
SNSTopic-1XYW5YD5V0M7C"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "ecs:RunTask"
            ],
```

```
            "Resource": [
                "arn:aws:ecs:ap-northeast-1:123456789012:task-definition/
FargateTaskNotification-ECSTaskDefinition-13YOJT8Z2LY5Q:1"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "ecs:StopTask",
                "ecs:DescribeTasks"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:ap-northeast-1:123456789012:rule/
StepFunctionsGetEventsForECSTaskRule"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Mentransfer catatan data (Lambda,DynamoDB,Amazon SQS)

Proyek sampel ini mendemonstrasikan cara membaca item secara iteratif dari Amazon DynamoDB tabel dan mengirim item ini ke Amazon SQS antrian menggunakan mesin status. Step Functions Menyebarkan proyek sampel ini akan membuat mesin Step Functions status, DynamoDB tabel, AWS Lambda fungsi, dan Amazon SQS antrian.

Dalam proyek ini, Step Functions menggunakan Lambda fungsi untuk mengisi DynamoDB tabel. Mesin state juga menggunakan `for` loop untuk membaca setiap entri, dan kemudian mengirim setiap entri ke Amazon SQS antrian.

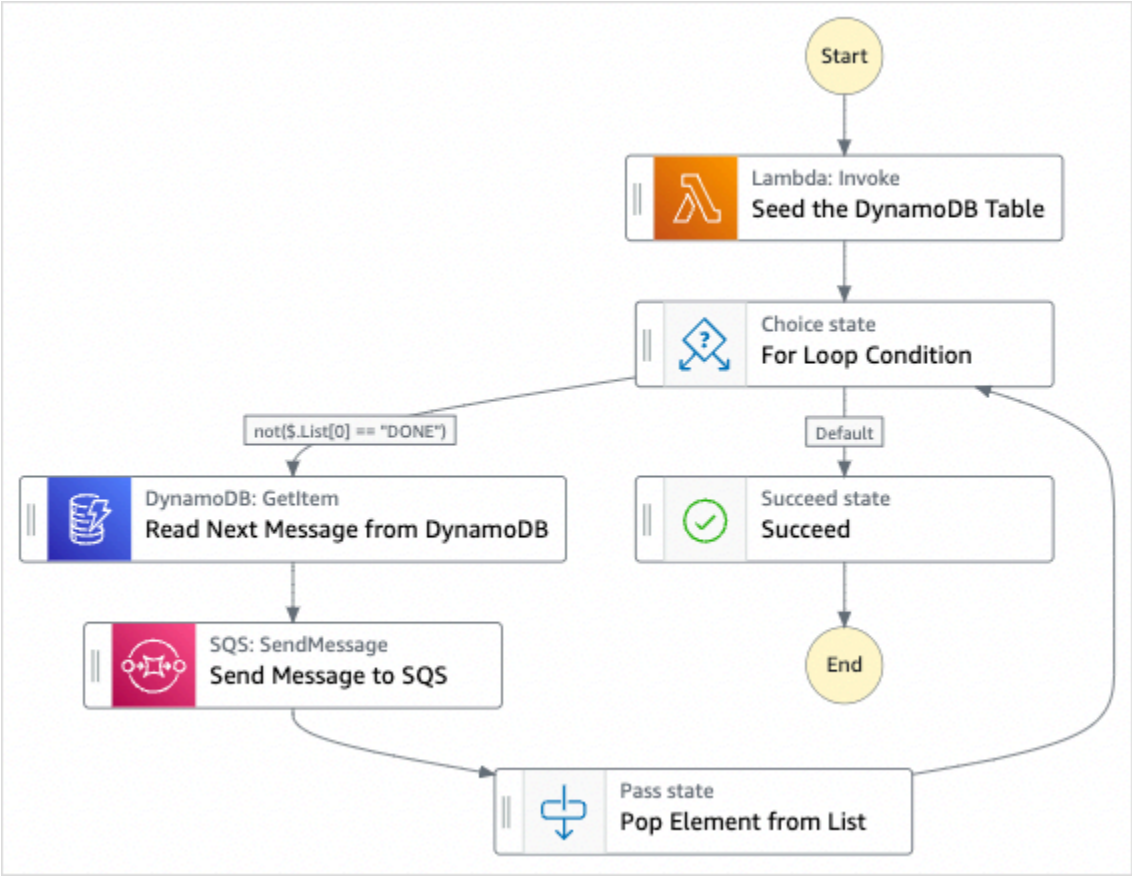
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`Transfer data records`** di kotak pencarian, lalu pilih Transfer catatan data dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

   Proyek contoh ini menyebarkan sumber daya berikut:

   - Sebuah fungsi Lambda untuk penyemaian tabel DynamoDB
   - Sebuah Antrean Amazon SQS
   - Sebuah Tabel DynamoDB
   - Mesin AWS Step Functions negara
   - Peran terkait AWS Identity and Access Management (IAM)

   Gambar berikut menunjukkan grafik alur kerja untuk proyek sampel catatan data Transfer:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

     Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

> **Important**
>
> Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

> **Tip**
>
> Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

> **Important**
>
> Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

1. Pada halaman mesin Negara, pilih proyek sampel Anda.
2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

   > **Note**
   >
   > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

   Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

   > **Note**
   >
   > Jika proyek demo yang Anda gunakan berisi data input eksekusi yang telah diisi sebelumnya, gunakan input tersebut untuk menjalankan mesin status.

3. Pilih Mulai Eksekusi.
4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

   Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Contoh Kode Mesin Status

Mesin status dalam proyek sampel ini terintegrasi dengan DynamoDB dan Amazon SQS dengan melewati parameter langsung ke sumber daya tersebut.

Jelajahi melalui contoh mesin status ini untuk melihat cara Step Functions mengontrol DynamoDB dan Amazon SQS dengan menghubungkan ke Amazon Resource Name (ARN) di bidang `Resource`, dan dengan melewati `Parameters` ke API layanan.

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

```
{
  "Comment" : "An example of the Amazon States Language for reading messages from a
 DynamoDB table and sending them to SQS",
  "StartAt": "Seed the DynamoDB Table",
  "TimeoutSeconds": 3600,
  "States": {
    "Seed the DynamoDB Table": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:sqsconnector-
SeedingFunction-T3U43VYDU5OQ",
      "ResultPath": "$.List",
      "Next": "For Loop Condition"
    },
    "For Loop Condition": {
      "Type": "Choice",
      "Choices": [
        {
          "Not": {
            "Variable": "$.List[0]",
            "StringEquals": "DONE"
          },
          "Next": "Read Next Message from DynamoDB"
        }
      ],
      "Default": "Succeed"
    },
    "Read Next Message from DynamoDB": {
      "Type": "Task",
      "Resource": "arn:aws:states:::dynamodb:getItem",
      "Parameters": {
        "TableName": "sqsconnector-DDBTable-1CAFOJWP8QD6I",
        "Key": {
          "MessageId": {"S.$": "$.List[0]"}
        }
      },
      "ResultPath": "$.DynamoDB",
```

```
      "Next": "Send Message to SQS"
    },
    "Send Message to SQS": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sqs:sendMessage",
      "Parameters": {
        "MessageBody.$": "$.DynamoDB.Item.Message.S",
        "QueueUrl": "https://sqs.us-east-1.amazonaws.com/123456789012/sqsconnector-
SQSQueue-QVGQBW134PWK"
      },
      "ResultPath": "$.SQS",
      "Next": "Pop Element from List"
    },
    "Pop Element from List": {
      "Type": "Pass",
      "Parameters": {
        "List.$": "$.List[1:]"
      },
      "Next": "For Loop Condition"
    },
    "Succeed": {
      "Type": "Succeed"
    }
  }
}
```

Untuk informasi selengkapnya tentang melewati parameter dan mengelola hasil, lihat berikut ini:

- Meneruskan parameter ke API layanan
- ResultPath

Contoh IAM

Contoh kebijakan AWS Identity and Access Management (IAM) yang dihasilkan oleh proyek sampel ini mencakup hak istimewa paling sedikit yang diperlukan untuk mengeksekusi mesin negara dan sumber daya terkait. Ini adalah praktik terbaik untuk menyertakan hanya izin yang diperlukan dalam kebijakan IAM Anda.

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

```
        {
            "Action": [
                "dynamodb:GetItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:ap-northeast-1:123456789012:table/
TransferDataRecords-DDBTable-3I41R5L5EAGT"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "sqs:SendMessage"
            ],
            "Resource": [
                "arn:aws:sqs:ap-northeast-1:123456789012:TransferDataRecords-SQSQueue-
BKWXTS09LIW1"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "lambda:invokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:ap-
northeast-1:123456789012:function:TransferDataRecords-SeedingFunction-VN4KY2TPAZSR"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Polling untuk Status Pekerjaan (Lambda,) AWS Batch

Proyek sampel ini menciptakan poller AWS Batch pekerjaan. Ini mengimplementasikan mesin AWS Step Functions status yang digunakan AWS Lambda untuk membuat loop `Wait` status yang memeriksa AWS Batch pekerjaan.

Proyek sampel ini membuat dan mengonfigurasi semua sumber daya sehingga alur kerja Step Functions Anda akan mengirimkan AWS Batch pekerjaan, dan akan menunggu pekerjaan itu selesai sebelum berakhir dengan sukses.

> **Note**
>
> Anda juga dapat menerapkan pola ini tanpa menggunakan fungsi Lambda. Untuk informasi tentang mengendalikan AWS Batch secara langsung, lihatMenggunakan AWS Step Functions dengan layanan lain.

Proyek sampel ini membuat mesin status, dua fungsi Lambda, dan AWS Batch antrian, dan mengonfigurasi izin IAM terkait.

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

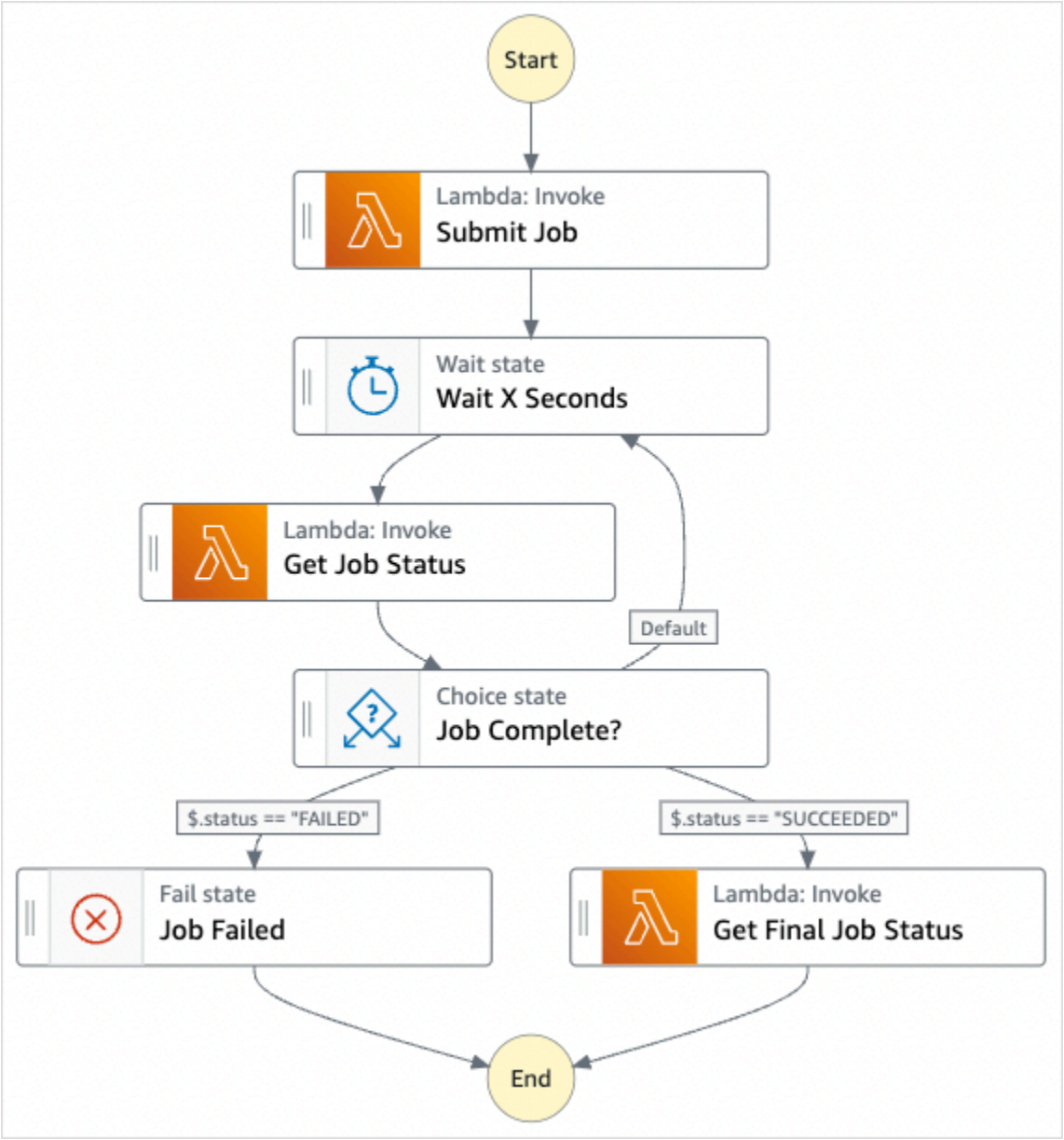
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`Job Poller`** di kotak pencarian, lalu pilih Job Poller dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

   Proyek contoh ini menyebarkan sumber daya berikut:

   - Tiga fungsi Lambda untuk mengirimkan AWS Batch pekerjaan, mendapatkan status saat ini dari AWS Batch pekerjaan yang diajukan, dan status penyelesaian pekerjaan akhir.
   - Sebuah AWS Batch pekerjaan
   - Mesin AWS Step Functions negara
   - Peran terkait AWS Identity and Access Management (IAM)

   Gambar berikut menunjukkan grafik alur kerja untuk proyek sampel Job Poller:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

     Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi

selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

> **Important**
>
> Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

  > **Tip**
  >
  > Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

  Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

  Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

  Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

  > **Important**
  >
  > Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

Setelah semua sumber daya disediakan dan digunakan, kotak dialog Mulai eksekusi ditampilkan dengan contoh input yang mirip dengan berikut ini.

```
{
    "jobName": "my-job",
    "jobDefinition": "arn:aws:batch:us-east-2:123456789012:job-definition/
SampleJobDefinition-343f54b445d5312:1",
    "jobQueue": "arn:aws:batch:us-east-2:123456789012:job-queue/
SampleJobQueue-4d9d696031e1449",
    "wait_time": 60
  }
```

> **Note**
>
> `wait_time` menginstruksikan status `Wait` untuk loop setiap 60 detik.

- Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

  1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

     > **Note**
     >
     > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

  2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

     Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

     > **Note**
     >
     > Jika proyek demo yang Anda gunakan berisi data input eksekusi yang telah diisi sebelumnya, gunakan input tersebut untuk menjalankan mesin status.

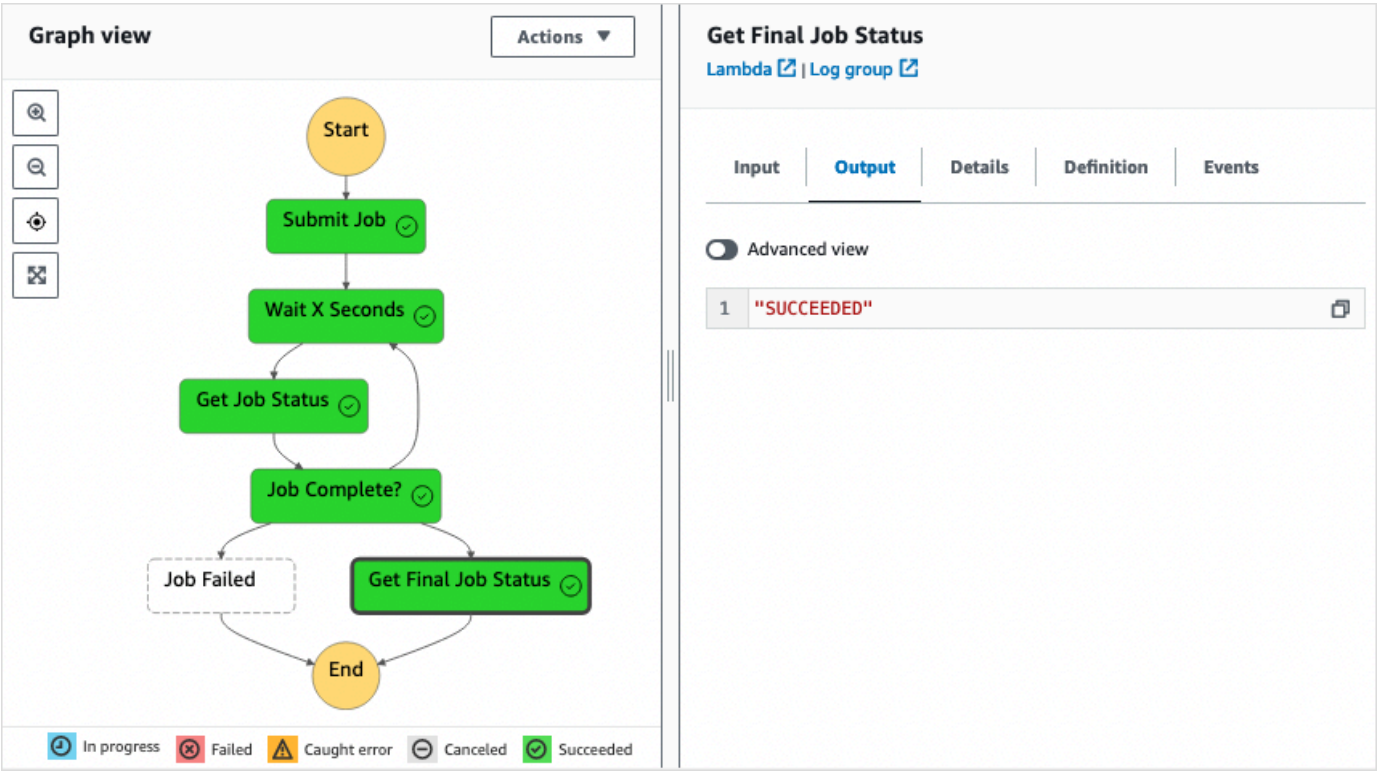
  3. Pilih Mulai Eksekusi.

4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

   Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Misalnya, untuk melihat perubahan status AWS Batch pekerjaan Anda dan hasil perulangan eksekusi Anda, pilih tab Output.

Gambar berikut menunjukkan grafik status eksekusi dalam tampilan Grafik. Ini juga menunjukkan output eksekusi untuk langkah yang dipilih di Output tab.



Contoh Kode Mesin Status

Mesin negara dalam proyek sampel ini terintegrasi dengan AWS Lambda untuk mengirimkan AWS Batch pekerjaan. Jelajahi contoh mesin status ini untuk melihat bagaimana Step Functions mengontrol Lambda dan. AWS Batch

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

```
{
    "Comment": "An example of the Amazon States Language that runs an AWS Batch job and
 monitors the job until it completes.",
    "StartAt": "Submit Job",
    "States": {
      "Submit Job": {
        "Type": "Task",
        "Resource": "arn:aws:lambda:us-
east-1:111122223333:function:StepFunctionsSample-JobStatusPol-SubmitJobFunction-
jDaYcl4cx55r",
        "ResultPath": "$.guid",
        "Next": "Wait X Seconds"
      },
      "Wait X Seconds": {
        "Type": "Wait",
        "SecondsPath": "$.wait_time",
        "Next": "Get Job Status"
      },
      "Get Job Status": {
        "Type": "Task",
        "Resource": "arn:aws:lambda:us-
east-1:111122223333:function:StepFunctionsSample-JobStatusPoll-
CheckJobFunction-1JkJwY10vonI",
        "Next": "Job Complete?",
        "InputPath": "$.guid",
        "ResultPath": "$.status"
      },
      "Job Complete?": {
        "Type": "Choice",
        "Choices": [
          {
            "Variable": "$.status",
            "StringEquals": "FAILED",
            "Next": "Job Failed"
          },
          {
            "Variable": "$.status",
            "StringEquals": "SUCCEEDED",
            "Next": "Get Final Job Status"
          }
```

```
        ],
        "Default": "Wait X Seconds"
      },
      "Job Failed": {
        "Type": "Fail",
        "Cause": "AWS Batch Job Failed",
        "Error": "DescribeJob returned FAILED"
      },
      "Get Final Job Status": {
        "Type": "Task",
        "Resource": "arn:aws::lambda:us-
east-1:111122223333:function:StepFunctionsSample-JobStatusPoll-
CheckJobFunction-1JkJwY10vonI",
        "InputPath": "$.guid",
        "End": true
      }
    }
  }
```

Timer Tugas (Lambda, Amazon SNS)

Proyek sampel ini membuat timer tugas. Ini mengimplementasikan mesin AWS Step Functions status yang mengimplementasikan `Wait` status, dan menggunakan AWS Lambda fungsi yang mengirimkan notifikasi Amazon Simple Notification Service (Amazon SNS). Status Tunggu adalah tipe status yang menunggu pemicu untuk melakukan satu unit kerja.

> **Note**
>
> Proyek contoh ini mengimplementasikan AWS Lambda fungsi untuk mengirim notifikasi Amazon Simple Notification Service (Amazon SNS). Anda juga dapat mengirim notifikasi Amazon SNS langsung dari Amazon States Language. Lihat Menggunakan AWS Step Functions dengan layanan lain.

Proyek sampel ini membuat mesin status, fungsi Lambda, dan topik Amazon SNS, dan mengonfigurasi izin AWS Identity and Access Management terkait (IAM). Untuk informasi selengkapnya tentang sumber daya yang dibuat dengan proyek sampel Timer Tugas, lihat berikut:

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

- AWS CloudFormation Panduan Pengguna
- Panduan Pengembang Layanan Pemberitahuan Sederhana Amazon
- AWS Lambda Panduan Pengembang
- Panduan Memulai IAM

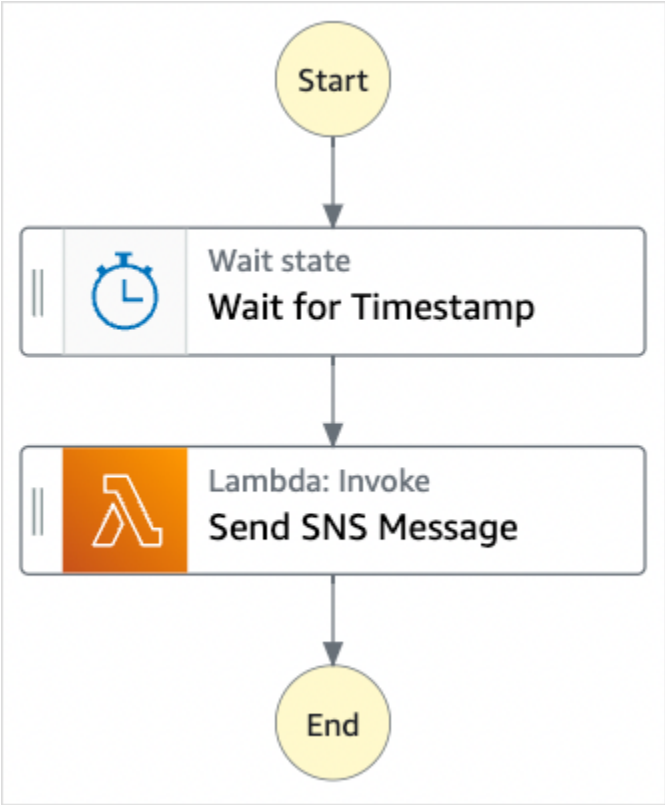
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`Task Timer`** di kotak pencarian, lalu pilih Pengatur Waktu Tugas dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

   Proyek contoh ini menyebarkan sumber daya berikut:

   - fungsi Lambda yang mengirimkan notifikasi Amazon SNS.
   - Mesin AWS Step Functions negara
   - Peran terkait AWS Identity and Access Management (IAM)

   Gambar berikut menunjukkan grafik alur kerja untuk proyek sampel Timer Tugas:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

     Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

     > **Important**
     >
     > Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

  > **Tip**
  >
  > Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

  Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

  Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

  Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

  > **Important**
  >
  > Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

Setelah semua sumber daya disediakan dan digunakan, kotak dialog Mulai eksekusi ditampilkan dengan contoh input yang mirip dengan berikut ini.

```
{
    "jobName": "my-job",{
  "topic": "arn:aws:sns:us-east-2:123456789012:StepFunctionsSample-TaskTimercc68840e-c3d3-42a8-911e-821b7ce248e5-SNSTopic-44UjcFxzhACT",
  "message": "HelloWorld",
  "timer_seconds": 10
}
```

- Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

   > **Note**
   >
   > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

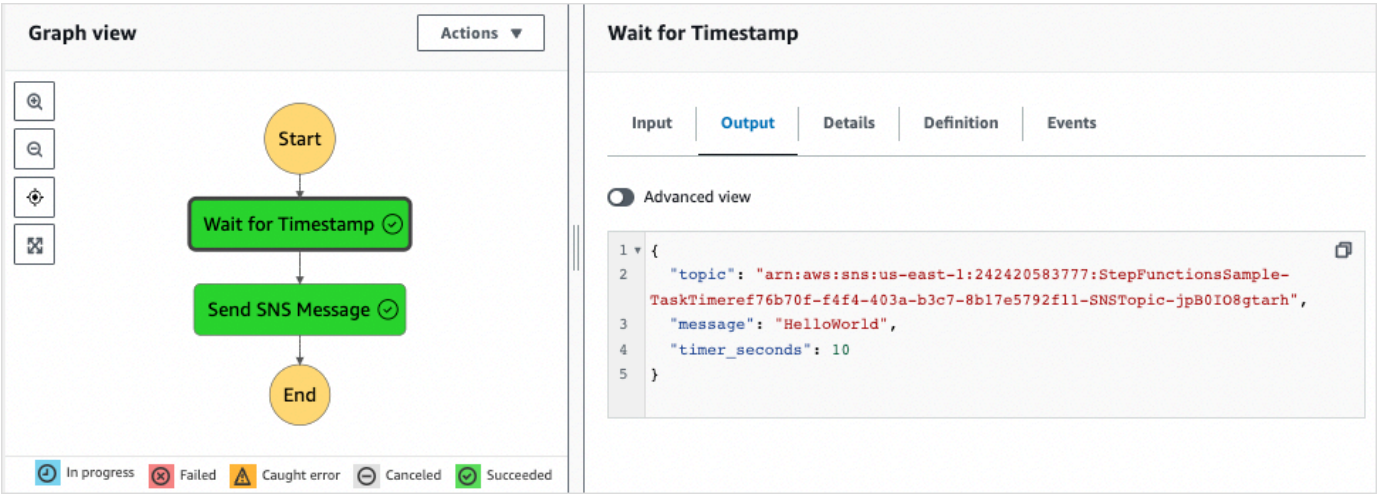
   Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

   > **Note**
   >
   > Jika proyek demo yang Anda gunakan berisi data input eksekusi yang telah diisi sebelumnya, gunakan input tersebut untuk menjalankan mesin status.

3. Pilih Mulai Eksekusi.
4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

   Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Misalnya, gambar berikut menunjukkan output dari langkah yang dipilih Tunggu Timestamp. Output dari langkah ini diteruskan sebagai input ke langkah Kirim Pesan SNS.



Contoh Pola Panggilan Balik (Amazon SQS, Amazon SNS, Lambda)

Proyek sampel ini menunjukkan cara AWS Step Functions jeda selama tugas, dan menunggu proses eksternal mengembalikan token tugas yang dihasilkan saat tugas dimulai.

Ketika proyek sampel ini di-deploy dan eksekusi dimulai, langkah-langkah berikut terjadi:

1. Step Functions meneruskan pesan yang mencakup token tugas untuk antrean Amazon Simple Queue Service (Amazon SQS).
2. Step Functions kemudian dijeda, menunggu token yang akan dikembalikan.
3. Antrian Amazon SQS memicu AWS Lambda fungsi yang memanggil `SendTaskSuccess`dengan token tugas yang sama.
4. Ketika token tugas diterima, alur kerja dilanjutkan.
5. Tugas `"Notify Success"` menerbitkan pesan Amazon Simple Notification Service (Amazon SNS) yang diterima panggilan balik.

Untuk mempelajari cara menerapkan pola panggilan balik di Step Functions, lihat Tunggu Panggilan Balik dengan Token Tugas.

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

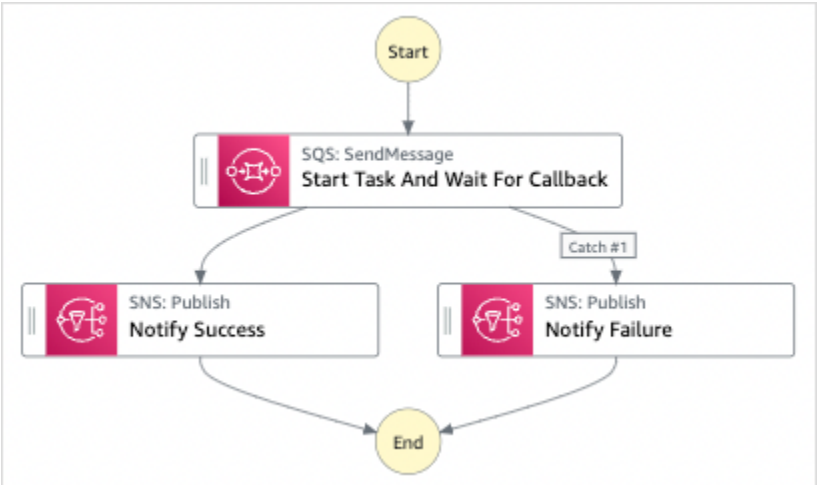
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`Callback pattern example`** di kotak pencarian, lalu pilih Contoh pola panggilan balik dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

   Proyek contoh ini menyebarkan sumber daya berikut:

   - Antrean pesan Amazon SQS.
   - Fungsi Lambda yang memanggil aksi Step Functions API. SendTaskSuccess
   - Topik Amazon SNS untuk memberi tahu tentang keberhasilan atau kegagalan tugas yang menunjukkan apakah alur kerja dapat berlanjut atau tidak.
   - Mesin AWS Step Functions negara
   - Peran terkait AWS Identity and Access Management (IAM)

   Gambar berikut menunjukkan grafik alur kerja untuk contoh proyek contoh pola Callback:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

- Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

  Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

  > **Important**
  >
  > Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

  > **Tip**
  >
  > Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

  Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

  Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

  Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

> **Important**
>
> Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

1. Pada halaman mesin Negara, pilih proyek sampel Anda.
2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

    1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

        > **Note**
        >
        > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

    2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

        Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

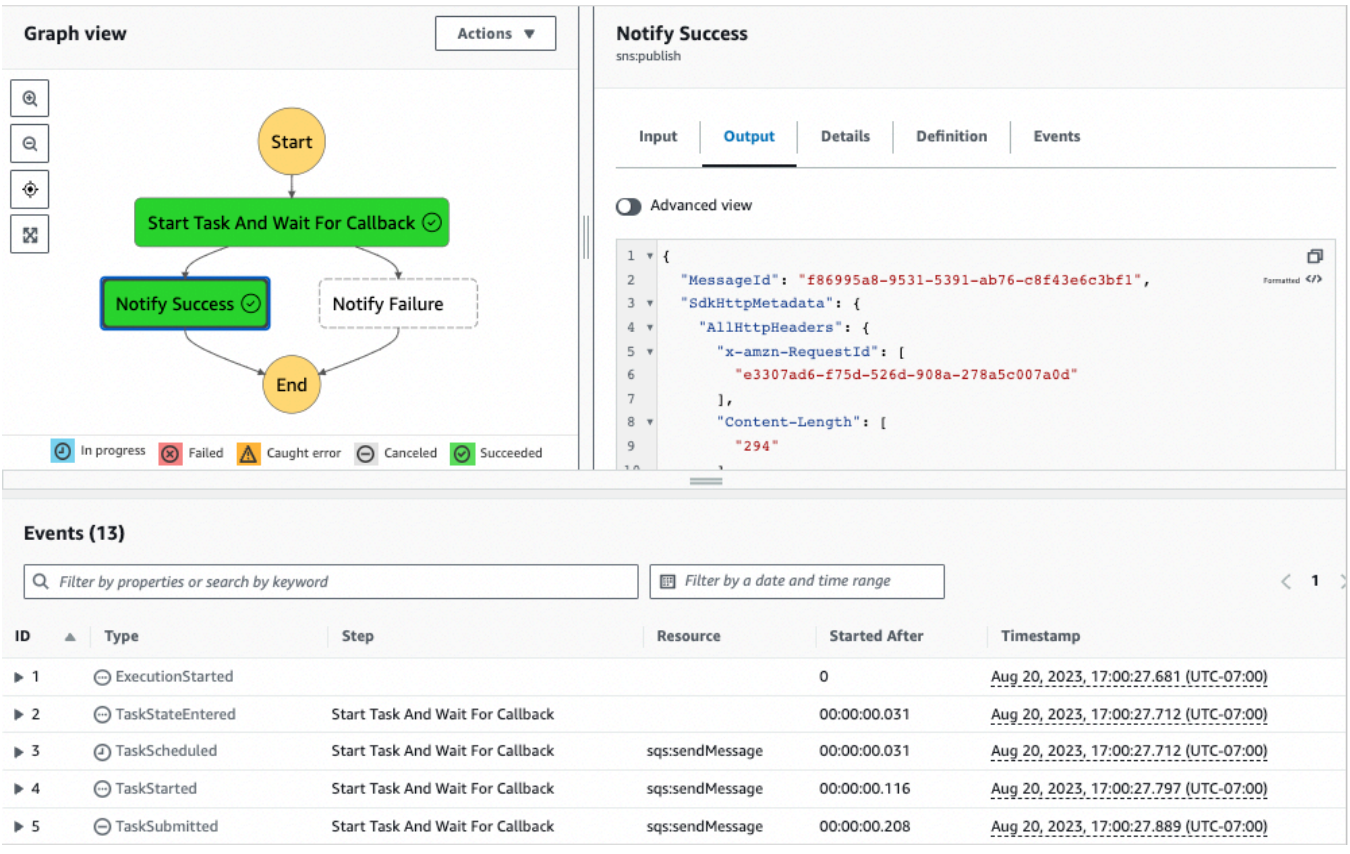
        > **Note**
        >
        > Jika proyek demo yang Anda gunakan berisi data input eksekusi yang telah diisi sebelumnya, gunakan input tersebut untuk menjalankan mesin status.

    3. Pilih Mulai Eksekusi.

4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

   Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Misalnya, untuk meninjau bagaimana Step Functions berkembang melalui alur kerja dan menerima panggilan balik dari Amazon SQS, tinjau entri dalam tabel Peristiwa. Gambar berikut menunjukkan output eksekusi untuk langkah Notify Success. Ini juga menunjukkan lima peristiwa pertama dari riwayat acara eksekusi. Perluas setiap acara untuk melihat detail lebih lanjut tentang acara itu.



Contoh Panggilan Balik Lambda

Untuk melihat bagaimana komponen proyek sampel ini bekerja sama, lihat sumber daya yang digunakan di AWS akun Anda. Misalnya, ini adalah fungsi Lambda yang memanggil Step Functions dengan token tugas.

```
console.log('Loading function');
const aws = require('aws-sdk');

exports.lambda_handler = (event, context, callback) => {
    const stepfunctions = new aws.StepFunctions();

    for (const record of event.Records) {
        const messageBody = JSON.parse(record.body);
        const taskToken = messageBody.TaskToken;

        const params = {
            output: "\"Callback task completed successfully.\"",
            taskToken: taskToken
        };

        console.log(`Calling Step Functions to complete callback task with params
 ${JSON.stringify(params)}`);

        stepfunctions.sendTaskSuccess(params, (err, data) => {
            if (err) {
                console.error(err.message);
                callback(err.message);
                return;
            }
            console.log(data);
            callback(null);
        });
    }
};
```

Kelola Tugas Amazon EMR

Proyek sampel ini menunjukkan EMR AWS Step Functions Amazon dan integrasi.

Ini menunjukkan cara untuk membuat klaster Amazon EMR, menambahkan beberapa langkah dan menjalankannya, lalu mengakhiri klaster.

> **Important**
>
> Amazon EMR tidak memiliki tingkat harga gratis. Menjalankan proyek sampel akan dikenakan biaya. Anda juga dapat menemukan informasi harga di halaman Harga Amazon EMR. Ketersediaan integrasi layanan Amazon EMR tergantung pada ketersediaan API Amazon EMR. Karena itu, proyek sampel ini mungkin tidak berfungsi dengan benar di beberapa AWS Wilayah. Lihat dokumentasi Amazon EMR untuk keterbatasan di Wilayah khusus.

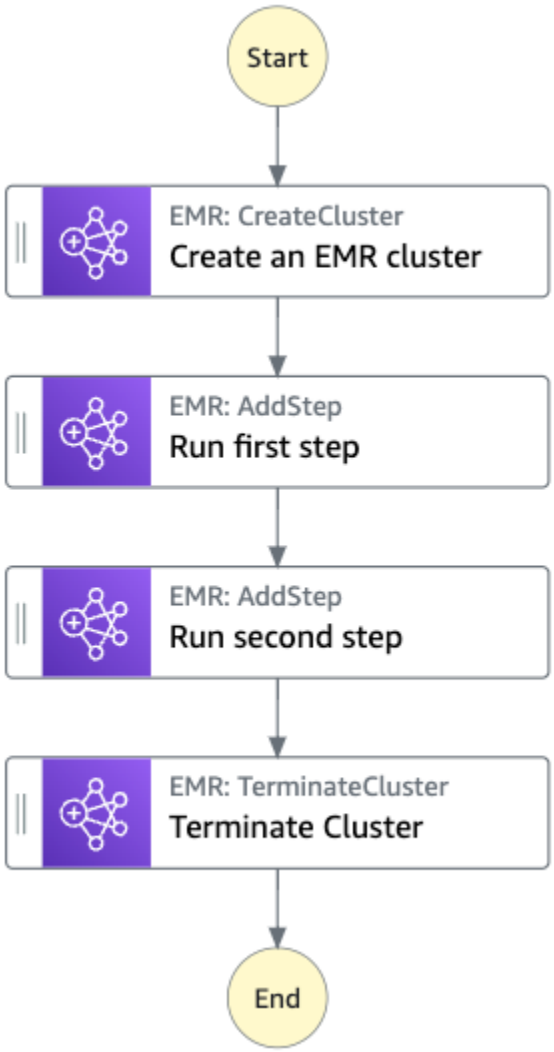
Langkah 1: Buat Mesin Negara dan Sumber Daya Penyediaan

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`Manage an EMR job`** di kotak pencarian, lalu pilih Kelola pekerjaan EMR dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

   Proyek contoh ini menyebarkan sumber daya berikut:

   - Amazon S3Ember
   - Sebuah Amazon EMR cluster
   - Mesin AWS Step Functions negara
   - Peran terkait AWS Identity and Access Management (IAM)

   Gambar berikut menunjukkan grafik alur kerja untuk Mengelola proyek sampel pekerjaan EMR:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

     Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

> **Important**
>
> Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

> **Tip**
>
> Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

> **Important**
>
> Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

1. Pada halaman mesin Negara, pilih proyek sampel Anda.
2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

   > **Note**
   >
   > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

   Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

   > **Note**
   >
   > Jika proyek demo yang Anda gunakan berisi data input eksekusi yang telah diisi sebelumnya, gunakan input tersebut untuk menjalankan mesin status.

3. Pilih Mulai Eksekusi.
4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

   Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Contoh Kode Mesin Status

Mesin status dalam proyek sampel ini terintegrasi dengan Amazon EMR dengan meneruskan parameter secara langsung ke sumber daya tersebut. Telusuri melalui mesin status contoh ini untuk

melihat cara Step Functions menggunakan mesin status untuk memanggil tugas Amazon EMR secara serentak, menunggu tugas untuk berhasil atau gagal, dan mengakhiri klaster.

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

```
{
  "Comment": "An example of the Amazon States Language for running jobs on Amazon EMR",
  "StartAt": "Create an EMR cluster",
  "States": {
    "Create an EMR cluster": {
      "Type": "Task",
      "Resource": "arn:<PARTITION>:states:::elasticmapreduce:createCluster.sync",
      "Parameters": {
        "Name": "ExampleCluster",
        "VisibleToAllUsers": true,
        "ReleaseLabel": "emr-5.26.0",
        "Applications": [
          { "Name": "Hive" }
        ],
        "ServiceRole": "<EMR_SERVICE_ROLE>",
        "JobFlowRole": "<EMR_EC2_INSTANCE_PROFILE>",
        "LogUri": "s3://<EMR_LOG_S3_BUCKET>/logs/",
        "Instances": {
          "KeepJobFlowAliveWhenNoSteps": true,
          "InstanceFleets": [
            {
              "Name": "MyMasterFleet",
              "InstanceFleetType": "MASTER",
              "TargetOnDemandCapacity": 1,
              "InstanceTypeConfigs": [
                {
                  "InstanceType": "m5.xlarge"
                }
              ]
            },
            {
              "Name": "MyCoreFleet",
              "InstanceFleetType": "CORE",
              "TargetOnDemandCapacity": 1,
              "InstanceTypeConfigs": [
                {
                  "InstanceType": "m5.xlarge"
```

```
              }
            ]
          }
        ]
      }
    },
    "ResultPath": "$.cluster",
    "Next": "Run first step"
  },
  "Run first step": {
    "Type": "Task",
    "Resource": "arn:<PARTITION>:states:::elasticmapreduce:addStep.sync",
    "Parameters": {
      "ClusterId.$": "$.cluster.ClusterId",
      "Step": {
        "Name": "My first EMR step",
        "ActionOnFailure": "CONTINUE",
        "HadoopJarStep": {
          "Jar": "command-runner.jar",
          "Args": ["<COMMAND_ARGUMENTS>"]
        }
      }
    },
    "Retry" : [
      {
        "ErrorEquals": [ "States.ALL" ],
        "IntervalSeconds": 1,
        "MaxAttempts": 3,
        "BackoffRate": 2.0
      }
    ],
    "ResultPath": "$.firstStep",
    "Next": "Run second step"
  },
  "Run second step": {
    "Type": "Task",
    "Resource": "arn:<PARTITION>:states:::elasticmapreduce:addStep.sync",
    "Parameters": {
      "ClusterId.$": "$.cluster.ClusterId",
      "Step": {
        "Name": "My second EMR step",
        "ActionOnFailure": "CONTINUE",
        "HadoopJarStep": {
          "Jar": "command-runner.jar",
```

```
            "Args": ["<COMMAND_ARGUMENTS>"]
          }
        }
      },
      "Retry" : [
        {
          "ErrorEquals": [ "States.ALL" ],
          "IntervalSeconds": 1,
          "MaxAttempts": 3,
          "BackoffRate": 2.0
        }
      ],
      "ResultPath": "$.secondStep",
      "Next": "Terminate Cluster"
    },
    "Terminate Cluster": {
      "Type": "Task",
      "Resource": "arn:<PARTITION>:states:::elasticmapreduce:terminateCluster",
      "Parameters": {
        "ClusterId.$": "$.cluster.ClusterId"
      },
      "End": true
    }
  }
}
```

Contoh IAM

Contoh kebijakan AWS Identity and Access Management (IAM) yang dihasilkan oleh proyek sampel ini mencakup hak istimewa paling sedikit yang diperlukan untuk mengeksekusi mesin negara dan sumber daya terkait. Ini adalah praktik terbaik untuk menyertakan hanya izin yang diperlukan dalam kebijakan IAM Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:RunJobFlow",
                "elasticmapreduce:DescribeCluster",
                "elasticmapreduce:TerminateJobFlows"
            ],
```

```
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "arn:aws:iam::123456789012:role/StepFunctionsSample-EMRJobManagement-
EMRServiceRole-ANPAJ2UCCR6DPCEXAMPLE",
                "arn:aws:iam::123456789012:role/StepFunctionsSample-
EMRJobManagementWJALRXUTNFEMI-ANPAJ2UCCR6DPCEXAMPLE-
EMREc2InstanceProfile-1ANPAJ2UCCR6DPCEXAMPLE"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:sa-east-1:123456789012:rule/
StepFunctionsGetEventForEMRRunJobFlowRule"
            ]
        }
    ]
}
```

Kebijakan berikut memastikan bahwa `addStep` memiliki izin yang memadai.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:AddJobFlowSteps",
                "elasticmapreduce:DescribeStep",
                "elasticmapreduce:CancelSteps"
            ],
            "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:sa-east-1:123456789012:rule/
StepFunctionsGetEventForEMRAddJobFlowStepsRule"
            ]
        }
    ]
}
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Jalankan EMR Serverless pekerjaan

Contoh proyek ini menunjukkan bagaimana Anda dapat membuat dan memulai EMR Serverless aplikasi. Proyek ini juga menunjukkan bagaimana Anda dapat menjalankan beberapa pekerjaan dalam aplikasi itu.

Proyek sampel ini membuat mesin status, AWS sumber daya pendukung, dan mengonfigurasi izin IAM terkait. Jelajahi proyek sampel ini untuk mempelajari tentang menjalankan EMR Serverless pekerjaan menggunakan mesin Step Functions negara, atau gunakan sebagai titik awal untuk proyek Anda sendiri.

> **Important**
>
> EMR Serverlesstidak memiliki tingkat harga gratis. Menjalankan proyek sampel akan dikenakan biaya. Anda dapat menemukan informasi harga di halaman Amazon EMR Serverlessharga.
> Selain itu, ketersediaan integrasi EMR Serverless layanan tergantung pada ketersediaan EMR Serverless API. Karena itu, proyek sampel ini mungkin tidak berfungsi dengan benar atau tersedia di beberapa proyek Wilayah AWS. Lihat topik Pertimbangan lain untuk informasi tentang ketersediaan EMR Serverless di Wilayah AWS.

Templat AWS CloudFormation dan sumber daya tambahan

Anda menggunakan CloudFormation template untuk menyebarkan proyek sampel ini. Template ini membuat sumber daya berikut di Akun AWS:

- Mesin Step Functions negara.
- Peran eksekusi untuk mesin negara. Peran ini memberikan izin yang dibutuhkan mesin status Anda untuk mengakses sumber lain Layanan AWS dan sumber daya seperti tindakan. EMR Serverless CreateApplication
- Peran eksekusi Job untukEMR Serverless. Peran ini memberikan izin yang dapat diasumsikan oleh EMR Serverless job run saat memanggil layanan lain atas nama Anda.

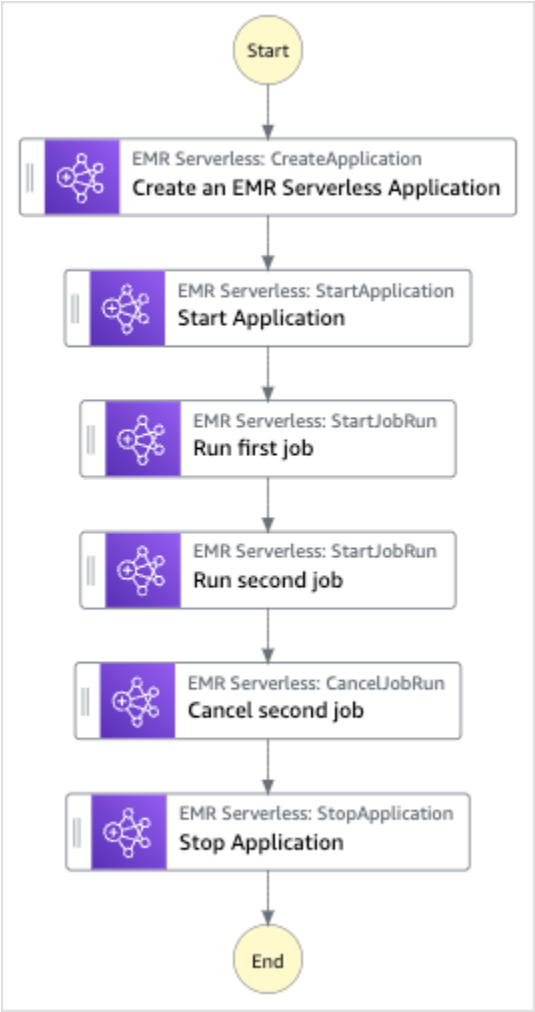
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`EMR Serverless`** di kotak pencarian, lalu pilih Jalankan EMR Serverless pekerjaan dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

   Proyek contoh ini menyebarkan sumber daya berikut:

   - Mesin Step Functions negara
   - Peran terkait AWS Identity and Access Management (IAM)

   Gambar berikut menunjukkan grafik alur kerja untuk proyek sampel Jalankan EMR Serverless pekerjaan:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

     Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

> **Important**
>
> Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

  > **Tip**
  >
  > Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

  Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

  Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

  Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

  > **Important**
  >
  > Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

1. Pada halaman mesin Negara, pilih proyek sampel Anda.
2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions secara otomatis menghasilkan nama eksekusi yang unik.

   > **Note**
   >
   > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, aktivitas, dan label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

   Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

3. Pilih Mulai Eksekusi.

4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

   Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Mulai Alur Kerja dalam Alur Kerja (Step Functions, Lambda)

Proyek sampel ini menunjukkan cara menggunakan mesin AWS Step Functions negara untuk memulai eksekusi mesin status lainnya. Untuk informasi tentang memulai eksekusi mesin status dari mesin negara lain, lihatMulai Eksekusi Alur Kerja dari Status Tugas..

Langkah 1: Buat mesin negara dan sumber daya penyediaan

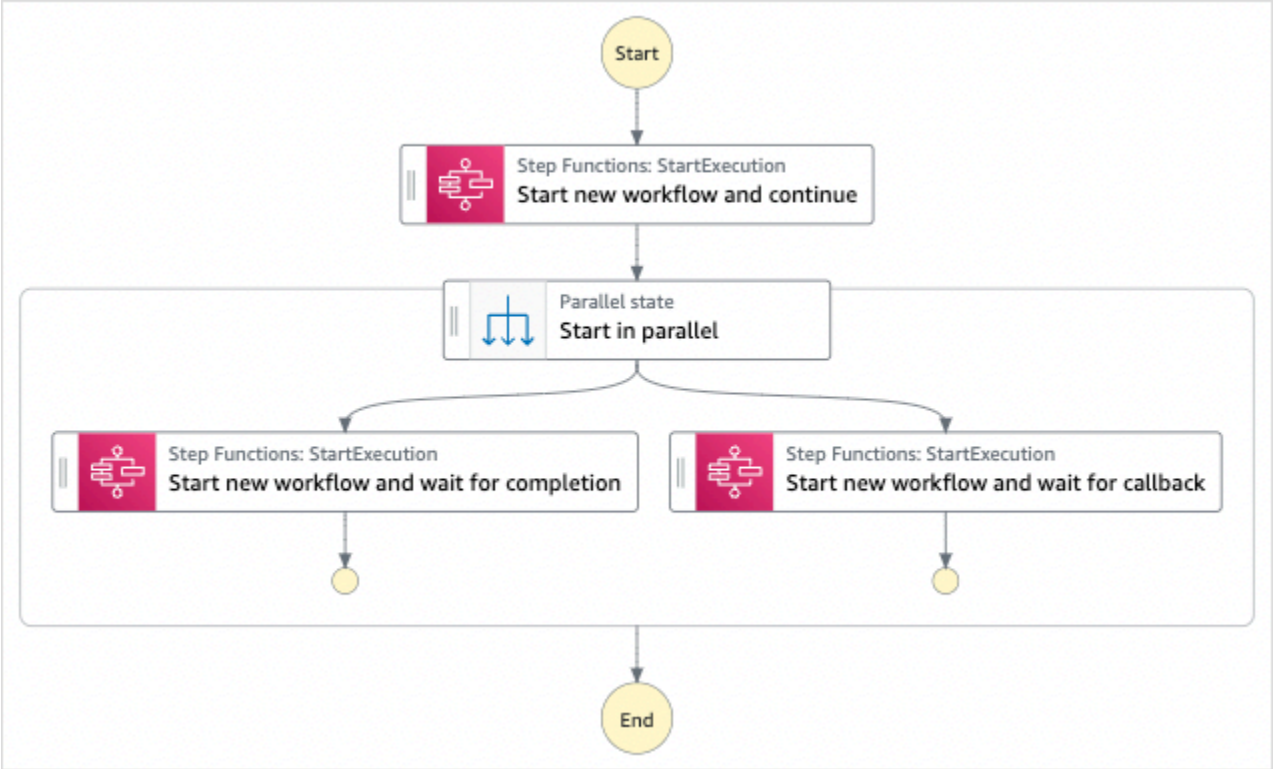
1. Buka Konsol Step Functions dan pilih Buat mesin status.

2. Ketik **`Start a workflow within a workflow`** kotak pencarian, lalu pilih Mulai alur kerja dalam alur kerja dari hasil penelusuran yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

   Proyek contoh ini menyebarkan sumber daya berikut:

   - Mesin negara tambahan. Eksekusi mesin state ini dimulai oleh mesin state yang Anda jalankan.
   - Fungsi callback Lambda. Fungsi ini digunakan dalam mesin status tambahan untuk mengimplementasikan mekanisme callback.
   - Mesin AWS Step Functions negara
   - Peran terkait AWS Identity and Access Management (IAM)

   Gambar berikut menunjukkan grafik alur kerja untuk Memulai alur kerja dalam proyek contoh alur kerja:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

     Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

     > **Important**
     >
     > Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

   - Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

     > **Tip**
     >
     > Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

     Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

     Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

     Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

> **Important**
>
> Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

1. Pada halaman mesin Negara, pilih proyek sampel Anda.
2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

    1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

        > **Note**
        >
        > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

    2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

        Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

        > **Note**
        >
        > Jika proyek demo yang Anda gunakan berisi data input eksekusi yang telah diisi sebelumnya, gunakan input tersebut untuk menjalankan mesin status.

    3. Pilih Mulai Eksekusi.

4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

   Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Contoh Kode Mesin Status

Mesin status dalam proyek sampel ini mengintegrasikan mesin status lain dan AWS Lambda dengan meneruskan parameter langsung ke sumber daya tersebut.

Jelajahi contoh mesin status ini untuk melihat bagaimana Step Functions memanggil tindakan API `StartExecution` untuk mesin status lainnya. Ini meluncurkan dua instans dari mesin status lainnya secara paralel: satu menggunakan pola Jalankan Tugas (.sync) dan satu menggunakan pola Tunggu Panggilan Balik dengan Token Tugas.

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

```
{
  "Comment": "An example of combining workflows using a Step Functions StartExecution
 task state with various integration patterns.",
  "StartAt": "Start new workflow and continue",
  "States": {
    "Start new workflow and continue": {
      "Comment": "Start an execution of another Step Functions state machine and
 continue",
      "Type": "Task",
      "Resource": "arn:aws:states:::states:startExecution",
      "Parameters": {
        "StateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:NestingPatternAnotherStateMachine-HZ9gtgspmdun",
        "Input": {
          "NeedCallback": false,
          "AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID.$": "$$.Execution.Id"
        }
      },
      "Next": "Start in parallel"
```

```
    },
    "Start in parallel": {
      "Comment": "Start two executions of the same state machine in parallel",
      "Type": "Parallel",
      "End": true,
      "Branches": [
        {
          "StartAt": "Start new workflow and wait for completion",
          "States": {
            "Start new workflow and wait for completion": {
              "Comment": "Start an execution of the same
 'NestingPatternAnotherStateMachine' and wait for its completion",
              "Type": "Task",
              "Resource": "arn:aws:states:::states:startExecution.sync",
              "Parameters": {
                "StateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:NestingPatternAnotherStateMachine-HZ9gtgspmdun",
                "Input": {
                  "NeedCallback": false,
                  "AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID.$": "$$.Execution.Id"
                }
              },
              "OutputPath": "$.Output",
              "End": true
            }
          }
        },
        {
          "StartAt": "Start new workflow and wait for callback",
          "States": {
            "Start new workflow and wait for callback": {
              "Comment": "Start an execution and wait for it to call back with a task
 token",
              "Type": "Task",
              "Resource": "arn:aws:states:::states:startExecution.waitForTaskToken",
              "Parameters": {
                "StateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:NestingPatternAnotherStateMachine-HZ9gtgspmdun",
                "Input": {
                  "NeedCallback": true,
                  "AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID.$": "$$.Execution.Id",
                  "TaskToken.$": "$$.Task.Token"
                }
              },
```

```
              "End": true
            }
          }
        }
      ]
    }
  }
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Secara dinamis memproses data dengan status Peta

Proyek sampel ini menunjukkan paralelisme dinamis menggunakan status Map.

Dalam proyek ini, Step Functions menggunakan AWS Lambda fungsi untuk menarik pesan dari antrian Amazon SQS, dan meneruskan array JSON dari pesan tersebut ke status. `Map` Untuk setiap pesan dalam antrean, mesin status menulis pesan untuk DynamoDB, memanggil fungsi Lambda lain untuk menghapus pesan dari Amazon SQS, lalu memublikasikan pesan untuk topik Amazon SNS.

Untuk informasi selengkapnya tentang status `Map` dan integrasi layanan Step Functions, lihat hal berikut:

- Map
- Menggunakan AWS Step Functions dengan layanan lain

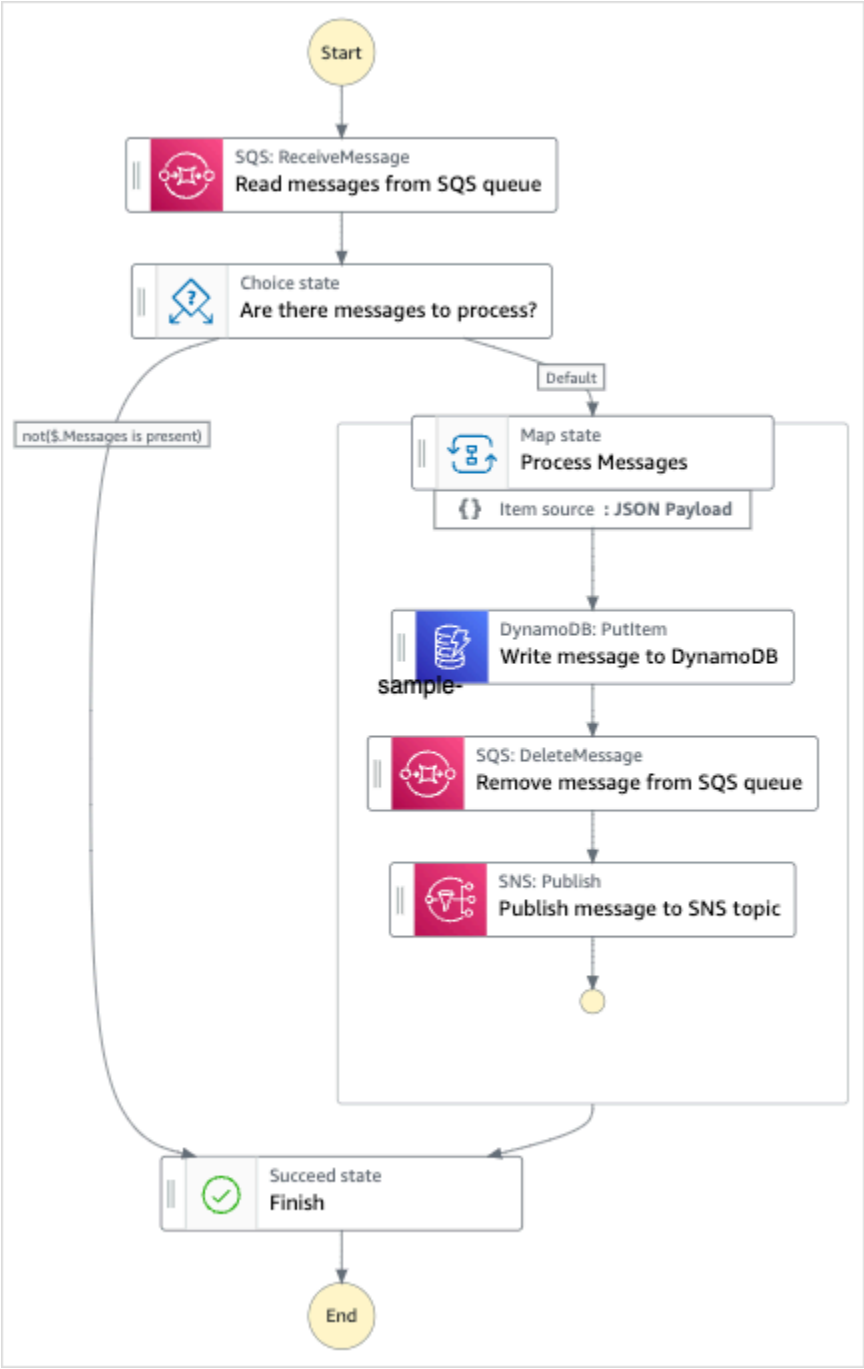
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`Dynamically process data with a Map state`** di kotak pencarian, lalu pilih Proses data secara dinamis dengan status Peta dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

Proyek contoh ini menyebarkan sumber daya berikut:

- Antrian Amazon SQS dari mana status Peta membaca dan menghapus pesan secara berulang.
- Sebuah tabel DynamoDB dimana status Peta menulis pesan secara iteratif.
- Topik Amazon SNS tempat Step Functions menerbitkan pesan yang dibacanya dari antrian Amazon SQS.
- Dua AWS Lambda fungsi
- Mesin AWS Step Functions negara
- Peran terkait AWS Identity and Access Management (IAM)

Gambar berikut menunjukkan grafik alur kerja untuk memproses data secara dinamis dengan proyek sampel status Peta:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

> **Important**
>
> Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

> **Tip**
>
> Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

> **Important**
>
> Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Setelah sumber daya proyek sampel diterapkan, Anda harus menambahkan item ke antrian Amazon SQS dan berlangganan topik Amazon SNS sebelum menjalankan mesin status.

Langkah 2: Berlangganan topik Amazon SNS

1. Buka konsol Amazon SNS.
2. Pilih Topik dan pilih topik yang dibuat oleh proyek sampel status `Map`.

   Nama akan mirip dengan MapSampleProj-snstopic-1cqo4hq3iR1kn.
3. Pilih Buat langganan.

   Halaman Buat langganan ditampilkan, mendaftarkan ARN Topik untuk topik.
4. Di Protokol, pilih Email.
5. Di Titik akhir, masukkan alamat email untuk berlangganan topik.
6. Pilih Buat Langganan.

   > **Note**
   >
   > Anda harus mengonfirmasi langganan di email Anda sebelum diaktifkan.

7. Buka email Konfirmasi langganan di akun terkait dan buka URL Konfirmasi langganan.

   Halaman Langganan terkonfirmasdi! ditampilkan.

Langkah 3: Tambahkan pesan ke antrian Amazon SQS

1. Buka konsol Amazon SQS.
2. Pilih antrean yang dibuat oleh proyek sampel status `Map`.

   Nama akan mirip dengan MapSampleProj-sqsqueue-1udic9vzdorn7.
3. Pilih Kirim dan terima pesan.
4. Pada halaman Kirim dan terima pesan, masukkan pesan dan pilih Kirim pesan.
5. Memasukkan pesan lain dan pilih Kirim pesan. Lanjutkan memasukkan lebih banyak pesan hingga Anda memiliki beberapa di antrian Amazon SQS.

Langkah 4: Jalankan mesin negara

> **Note**
>
> Antrean di Amazon SNS akhirnya konsisten. Untuk hasil terbaik, tunggu beberapa menit saat mengisi antrean hingga menjalankan eksekusi mesin status Anda.

1. Pada halaman mesin Negara, pilih proyek sampel Anda.
2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

    1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

        > **Note**
        >
        > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

    2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

        Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

        > **Note**
        >
        > Jika proyek demo yang Anda gunakan berisi data input eksekusi yang telah diisi sebelumnya, gunakan input tersebut untuk menjalankan mesin status.

    3. Pilih Mulai Eksekusi.
    4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Contoh kode mesin status

Mesin status dalam proyek sampel ini terintegrasi dengan Amazon SQS, Amazon SNS, dan Lambda dengan meneruskan parameter secara langsung ke sumber daya tersebut.

Jelajahi mesin status contoh ini untuk melihat bagaimana Step Functions mengendalikan Lambda, DynamoDB, Amazon SNS dengan menghubungkan ke Amazon Resource Name (ARN) di bidang `Resource`, dan dengan meneruskan `Parameters` ke API layanan.

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

```
{
  "Comment": "An example of the Amazon States Language for reading messages from an SQS
 queue and iteratively processing each message.",
  "StartAt": "Read messages from SQS Queue",
  "States": {
    "Read messages from SQS Queue": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "OutputPath": "$.Payload",
      "Parameters": {
        "FunctionName": "MapSampleProj-ReadFromSQSQueueLambda-1MY3M63RMJVA9"
      },
      "Next": "Are there messages to process?"
    },
    "Are there messages to process?": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$",
          "StringEquals": "No messages",
          "Next": "Finish"
        }
      ],
      "Default": "Process messages"
```

```
    },
    "Process messages": {
      "Type": "Map",
      "Next": "Finish",
      "ItemsPath": "$",
      "Parameters": {
        "MessageNumber.$": "$$.Map.Item.Index",
        "MessageDetails.$": "$$.Map.Item.Value"
      },
      "Iterator": {
        "StartAt": "Write message to DynamoDB",
        "States": {
          "Write message to DynamoDB": {
            "Type": "Task",
            "Resource": "arn:aws:states:::dynamodb:putItem",
            "ResultPath": null,
            "Parameters": {
              "TableName": "MapSampleProj-DDBTable-YJDJ1MKIN6C5",
              "ReturnConsumedCapacity": "TOTAL",
              "Item": {
                "MessageId": {
                  "S.$": "$.MessageDetails.MessageId"
                },
                "Body": {
                  "S.$": "$.MessageDetails.Body"
                }
              }
            },
            "Next": "Remove message from SQS queue"
          },
          "Remove message from SQS queue": {
            "Type": "Task",
            "Resource": "arn:aws:states:::lambda:invoke",
            "InputPath": "$.MessageDetails",
            "ResultPath": null,
            "Parameters": {
              "FunctionName": "MapSampleProj-DeleteFromSQSQueueLambda-198J2839ZO5K2",
              "Payload": {
                "ReceiptHandle.$": "$.ReceiptHandle"
              }
            },
            "Next": "Publish message to SNS topic"
          },
          "Publish message to SNS topic": {
```

```
            "Type": "Task",
            "Resource": "arn:aws:states:::sns:publish",
            "InputPath": "$.MessageDetails",
            "Parameters": {
              "Subject": "Message from Step Functions!",
              "Message.$": "$.Body",
              "TopicArn": "arn:aws:sns:us-east-1:012345678910:MapSampleProj-
SNSTopic-1CQO4HQ3IR1KN"
            },
            "End": true
          }
        }
      }
    },
    "Finish": {
      "Type": "Succeed"
    }
  }
}
```

Contoh IAM

Contoh kebijakan AWS Identity and Access Management (IAM) yang dihasilkan oleh proyek sampel ini mencakup hak istimewa paling sedikit yang diperlukan untuk mengeksekusi mesin negara dan sumber daya terkait. Kami merekomendasikan agar Anda hanya menyertakan izin yang diperlukan dalam kebijakan IAM Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:012345678901:function:MapSampleProj-
ReadFromSQSQueueLambda-1MY3M63RMJVA9",
                "arn:aws:lambda:us-east-1:012345678901:function:MapSampleProj-
DeleteFromSQSQueueLambda-198J2839ZO5K2"
            ],
            "Effect": "Allow"
        },
        {
```

```
            "Action": [
                "dynamodb:PutItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-east-1:012345678901:table/MapSampleProj-DDBTable-
YJDJ1MKIN6C5"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:us-east-1:012345678901:MapSampleProj-
SNSTopic-1CQO4HQ3IR1KN"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Memproses file CSV dengan Peta Terdistribusi

Proyek contoh ini menunjukkan bagaimana Anda dapat menggunakan status Peta Terdistribusi untuk mengulangi lebih dari 10.000 baris file CSV yang dihasilkan menggunakan fungsi. Lambda File CSV berisi informasi pengiriman pesanan pelanggan dan disimpan dalam bucket Amazon S3. Peta Terdistribusi mengulangi lebih dari 10 baris dalam file CSV untuk analisis data.

Peta Terdistribusi berisi Lambda fungsi untuk mendeteksi pesanan yang tertunda. Peta Terdistribusi juga berisi Peta Inline untuk memproses pesanan tertunda dalam batch dan mengembalikan pesanan tertunda ini dalam sebuah array. Untuk setiap order yang tertunda, Inline Map mengirimkan pesan ke Amazon SQS antrian. Terakhir, proyek contoh ini menyimpan hasil Map Run ke bucket Amazon S3 lain di bucket Anda. Akun AWS

Dengan Distributed Map, Anda dapat menjalankan hingga 10.000 eksekusi alur kerja anak paralel sekaligus. Dalam proyek sampel ini, konkurensi maksimum Peta Terdistribusi ditetapkan pada 1000 yang membatasi hingga 1000 eksekusi alur kerja anak paralel.

Proyek sampel ini membuat mesin status, AWS sumber daya pendukung, dan mengonfigurasi izin IAM terkait. Jelajahi proyek sampel ini untuk mempelajari tentang menggunakan Peta Terdistribusi untuk mengatur beban kerja paralel skala besar, atau menggunakannya sebagai titik awal untuk proyek Anda sendiri.

AWS CloudFormation template dan sumber daya tambahan

Anda menggunakan CloudFormation template untuk menyebarkan proyek sampel ini. Template ini membuat sumber daya berikut di Akun AWS:

- Mesin status Step Functions.
- Peran eksekusi untuk mesin negara. Peran ini memberikan izin yang dibutuhkan mesin status Anda untuk mengakses sumber lain Layanan AWS dan sumber daya seperti tindakan Invoke fungsi Lambda.
- Fungsi Lambda bernama `CSVGeneratorFunction` yang menghasilkan file CSV yang berisi detail pesanan pelanggan.
- Peran eksekusi untuk fungsi Lambda generator CSV. Peran ini memberikan izin fungsi untuk mengakses lainnya Layanan AWS.
- Bucket input Amazon S3 untuk menyimpan file CSV yang dihasilkan.
- Fungsi Lambda deteksi pesanan tertunda yang menganalisis data file CSV dan mendeteksi pesanan yang tertunda.
- Peran eksekusi untuk fungsi Lambda order tertunda. Peran ini memberikan izin fungsi untuk mengakses lainnya Layanan AWS.
- Bucket keluaran Amazon S3 untuk menyimpan hasil analisis pesanan pelanggan.
- Antrian Amazon SQS tempat Step Functions mengirim pesan untuk setiap pesanan yang tertunda. Pesan-pesan ini berisi ID pelanggan dan pesanan mereka.
- Grup CloudWatch log yang menyimpan informasi yang terkait dengan riwayat eksekusi mesin negara.

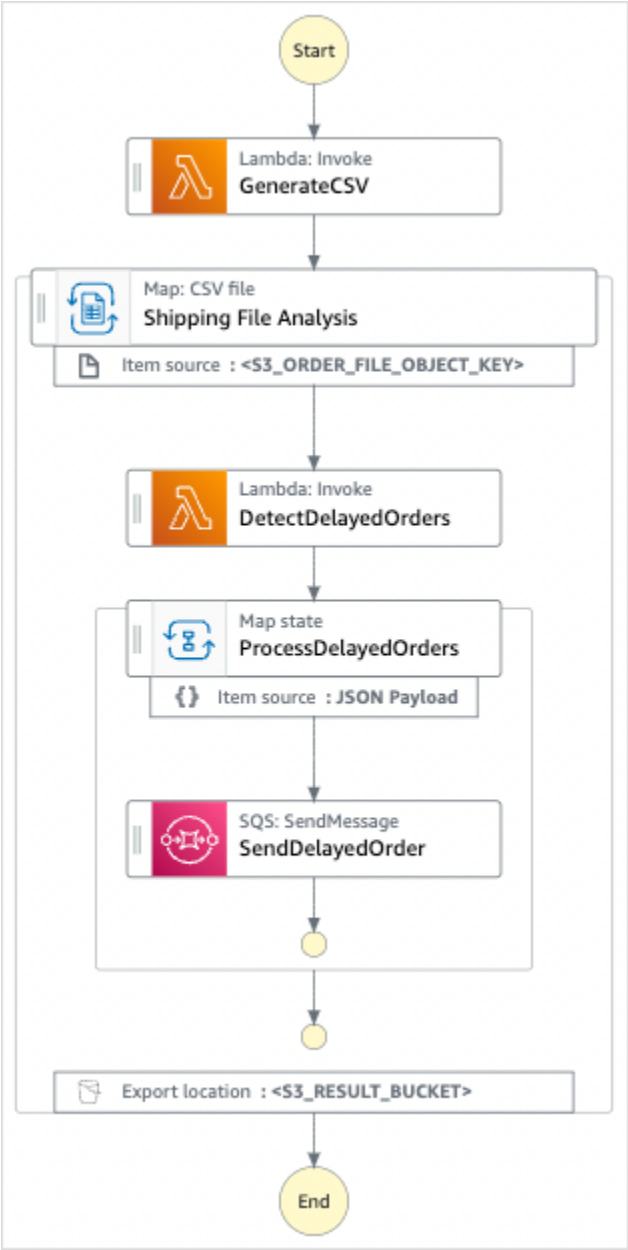
> **Important**
>
> Biaya standar berlaku untuk setiap layanan.

Langkah 1: Buat mesin negara dan sumber daya penyediaan

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`Distributed Map to process a CSV file in S3`** di kotak pencarian, lalu pilih Peta Terdistribusi untuk memproses file CSV di S3 dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

   Untuk informasi tentang sumber daya yang akan dibuat untuk proyek sampel ini, lihatAWS CloudFormation template dan sumber daya tambahan.

   Gambar berikut menunjukkan grafik alur kerja untuk Peta Terdistribusi untuk memproses file CSV dalam proyek sampel S3:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu dari cara berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

     Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon

States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

> **Important**
>
> Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

  > **Tip**
  >
  > Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

  Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

  Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

  Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

  > **Important**
  >
  > Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

Setelah semua sumber daya disediakan dan digunakan, Anda dapat menjalankan mesin status.

1. Pada halaman mesin Negara, pilih proyek sampel Anda.
2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

   a. (Opsional) Masukkan nilai input dalam format JSON untuk menjalankan proyek sampel Anda.

      Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

      > **Note**
      >
      > Jika proyek demo yang Anda gunakan berisi data input eksekusi yang telah diisi sebelumnya, gunakan input tersebut untuk menjalankan mesin status.

   b. Pilih Mulai Eksekusi.

   c. (Opsional) Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

      Setelah eksekusi selesai, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap negara bagian termasuk input, output, dan definisi masing-masing.

      - Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.
      - Untuk informasi selengkapnya tentang melihat eksekusi status Peta Terdistribusi di konsol, lihatMemeriksa Map Run.

   d. (Opsional) Tinjau hasil eksekusi yang diekspor ke bucket Amazon S3. Hasil ini termasuk data, seperti input dan output eksekusi, ARN, dan status eksekusi. Lihat informasi yang lebih lengkap di ResultWriter.

Memproses data dalam bucket Amazon S3 dengan Peta Terdistribusi

Proyek sampel ini menunjukkan bagaimana Anda dapat menggunakan status Peta Terdistribusi untuk memproses data skala besar, misalnya, menganalisis data cuaca historis dan mengidentifikasi stasiun cuaca yang memiliki suhu rata-rata tertinggi di planet ini setiap bulan. Data cuaca direkam di lebih dari 12.000 file CSV, yang pada gilirannya disimpan dalam ember Amazon S3.

Proyek sampel ini mencakup dua status Peta Terdistribusi bernama Salinan S3 Terdistribusi NOA Data dan ProcessNoAadata. Salinan S3 terdistribusi NOA Data iterasi melalui file CSV dalam bucket Amazon S3 publik bernama noaa-gsod-pdsdan menyalinnya ke bucket Amazon S3 di bucket Anda. Akun AWS ProcessNoAadata mengulangi file yang disalin dan menyertakan fungsi Lambda yang melakukan analisis suhu.

Proyek sampel pertama-tama memeriksa konten bucket Amazon S3 dengan panggilan ke tindakan ListObjectsV2 API. Berdasarkan jumlah kunci yang dikembalikan sebagai tanggapan atas panggilan ini, proyek sampel mengambil salah satu keputusan berikut:

- Jika jumlah kunci lebih dari atau sama dengan 1, proyek transisi ke status processNoAadata. Status Peta Terdistribusi ini mencakup Lambda fungsi bernama TemperatureFunctionyang menemukan stasiun cuaca yang memiliki suhu rata-rata tertinggi setiap bulan. Fungsi ini mengembalikan kamus dengan `year-month` sebagai kunci dan kamus yang berisi informasi tentang stasiun cuaca sebagai nilai.
- Jika jumlah kunci yang dikembalikan tidak melebihi 1, status Data NOA Salin S3 Terdistribusi akan mencantumkan semua objek dari bucket publik noaa-gsod-pdsdan secara berulang menyalin objek individual ke bucket lain di akun Anda dalam batch 100. Peta Inline melakukan penyalinan objek secara berulang.

  Setelah semua objek disalin, proyek bertransisi ke status processNoAAdata untuk memproses data cuaca.

Proyek sampel akhirnya beralih ke Lambda fungsi peredam yang melakukan agregasi akhir dari hasil yang dikembalikan oleh TemperatureFunctionfungsi dan menulis hasilnya ke tabel. Amazon DynamoDB

Dengan Distributed Map, Anda dapat menjalankan hingga 10.000 eksekusi alur kerja anak paralel sekaligus. Dalam proyek sampel ini, konkurensi maksimum ProcessNoAADATA Distributed Map ditetapkan pada 3000 yang membatasi hingga 3000 eksekusi alur kerja anak paralel.

Proyek sampel ini membuat mesin status, AWS sumber daya pendukung, dan mengonfigurasi izin IAM terkait. Jelajahi proyek sampel ini untuk mempelajari tentang menggunakan Peta Terdistribusi untuk mengatur beban kerja paralel skala besar, atau menggunakannya sebagai titik awal untuk proyek Anda sendiri.

> **Important**
>
> Proyek sampel ini hanya tersedia di Wilayah AS Timur (Virginia N.).

AWS CloudFormation template dan sumber daya tambahan

Anda menggunakan CloudFormation template untuk menyebarkan proyek sampel ini. Template ini membuat sumber daya berikut di Akun AWS:

- Mesin status Step Functions.
- Peran eksekusi untuk mesin negara. Peran ini memberikan izin yang dibutuhkan mesin status Anda untuk mengakses sumber lain Layanan AWS dan sumber daya seperti tindakan Invoke fungsi Lambda.
- Ember Amazon S3 bernama. `N0AADataBucket` Bucket ini berisi file CSV dengan data cuaca.
- Fungsi Lambda bernama `ReducerFunction` yang melakukan agregasi akhir data cuaca dan menulis hasilnya ke tabel Amazon DynamoDB.
- Peran eksekusi untuk fungsi Lambda peredam. Peran ini memberikan izin fungsi untuk mengakses lainnya Layanan AWS.
- Bucket keluaran Amazon S3 diberi nama `ResultsBucket` untuk menyimpan hasil analisis cuaca.
- Sebuah tabel DynamoDB `ResultsDynamoDBTable` bernama yang berisi hasil yang dikembalikan oleh. `ReducerFunction`
- Fungsi Lambda bernama `TemperatureFunction` yang menemukan suhu rata-rata bulanan tertinggi.
- Peran eksekusi untuk fungsi Lambda. Peran ini memberikan izin fungsi untuk mengakses lainnya Layanan AWS.

- Grup CloudWatch log yang menyimpan informasi yang terkait dengan riwayat eksekusi mesin negara.

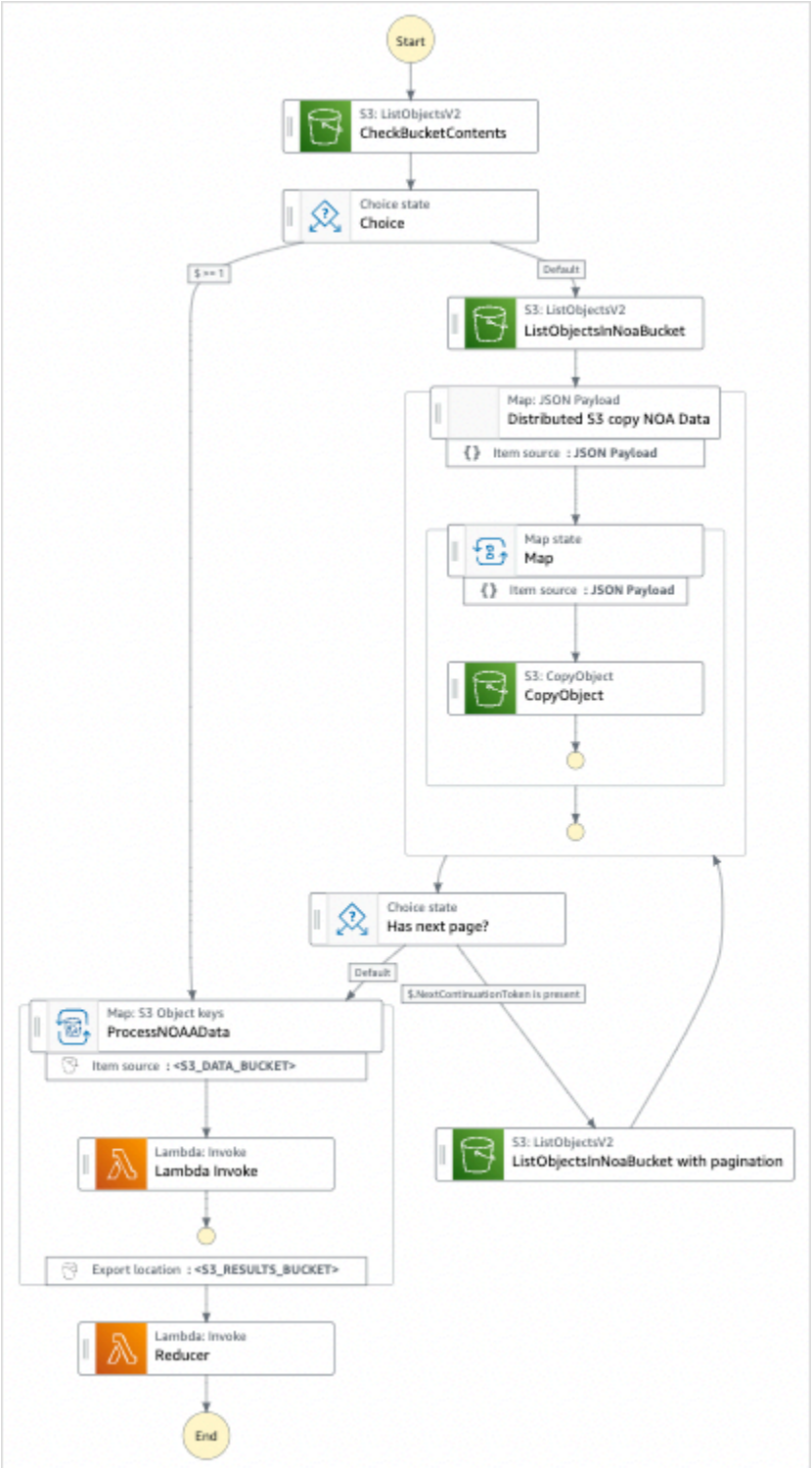
> **Important**
>
> Biaya standar berlaku untuk setiap layanan.

Langkah 1: Buat mesin negara dan sumber daya penyediaan

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`Distributed Map to process files in S3`** di kotak pencarian, lalu pilih Peta Terdistribusi untuk memproses file di S3 dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

   Untuk informasi tentang sumber daya yang akan dibuat untuk proyek sampel ini, lihatAWS CloudFormation template dan sumber daya tambahan.

   Gambar berikut menunjukkan grafik alur kerja untuk Peta Terdistribusi untuk memproses file dalam proyek sampel S3:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu dari cara berikut:

- Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

  Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

  > **Important**
  >
  > Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

  > **Tip**
  >
  > Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

  Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

  Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

  Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

> **Important**
>
> Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

Setelah semua sumber daya disediakan dan digunakan, Anda dapat menjalankan mesin status.

1. Pada halaman mesin Negara, pilih proyek sampel Anda.
2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

   a. (Opsional) Masukkan nilai input dalam format JSON untuk menjalankan proyek sampel Anda.

      Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

      > **Note**
      >
      > Jika proyek demo yang Anda gunakan berisi data input eksekusi yang telah diisi sebelumnya, gunakan input tersebut untuk menjalankan mesin status.

   b. Pilih Mulai Eksekusi.

   c. (Opsional) Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

      Setelah eksekusi selesai, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap negara bagian termasuk input, output, dan definisi masing-masing.

      - Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

- Untuk informasi selengkapnya tentang melihat eksekusi status Peta Terdistribusi di konsol, lihatMemeriksa Map Run.

d. (Opsional) Tinjau hasil eksekusi yang diekspor ke bucket Amazon S3. Hasil ini termasuk data, seperti input dan output eksekusi, ARN, dan status eksekusi. Lihat informasi yang lebih lengkap di ResultWriter.

Melatih Model Machine Learning

Proyek sampel ini menunjukkan cara menggunakan SageMaker dan AWS Step Functions melatih model pembelajaran mesin dan cara mengubah kumpulan data pengujian secara batch.

Dalam proyek ini, Step Functions menggunakan fungsi Lambda untuk menempatkan bucket Amazon S3 dengan set data uji. Kemudian melatih model pembelajaran mesin dan melakukan transformasi batch, menggunakan integrasi SageMaker layanan.

Untuk informasi selengkapnya tentang SageMaker integrasi layanan Step Functions, lihat berikut ini:

- Menggunakan AWS Step Functions dengan layanan lain
- Kelola SageMaker dengan Step Functions

> **Note**
>
> Proyek sampel ini mungkin dikenakan biaya.
> Untuk AWS pengguna baru, tingkat penggunaan gratis tersedia. Pada tingkat ini, layanan akan gratis di bawah tingkat penggunaan tertentu. Untuk informasi selengkapnya tentang AWS biaya dan Tingkat Gratis, lihat SageMaker Harga.

Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan

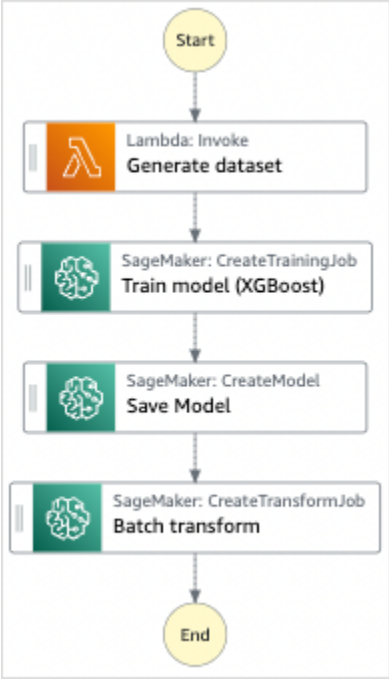
1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`Train a machine learning model`** di kotak pencarian, lalu pilih Latih model pembelajaran mesin dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke

Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

Proyek contoh ini menyebarkan sumber daya berikut:

- Sebuah AWS Lambda fungsi
- Bucket Amazon Simple Storage Service (Amazon S3)
- Mesin AWS Step Functions negara
- Peran terkait AWS Identity and Access Management (IAM)

Gambar berikut menunjukkan grafik alur kerja untuk proyek sampel model pembelajaran mesin Latih:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

     Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon

States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

> **Important**
>
> Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

  > **Tip**
  >
  > Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

  Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

  Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

  Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

  > **Important**
  >
  > Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

1. Pada halaman mesin Negara, pilih proyek sampel Anda.

2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

    1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

        > **Note**
        >
        > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

    2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

        Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

        > **Note**
        >
        > Jika proyek demo yang Anda gunakan berisi data input eksekusi yang telah diisi sebelumnya, gunakan input tersebut untuk menjalankan mesin status.

    3. Pilih Mulai Eksekusi.
    4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

        Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Contoh Kode Mesin Status

Mesin status dalam proyek sampel ini terintegrasi dengan SageMaker dan AWS Lambda dengan meneruskan parameter langsung ke sumber daya tersebut, dan menggunakan bucket Amazon S3 untuk sumber dan output data pelatihan.

Jelajahi contoh mesin status ini untuk melihat bagaimana Step Functions mengontrol Lambda dan. SageMaker

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

```
{
  "StartAt": "Generate dataset",
  "States": {
    "Generate dataset": {
      "Resource": "arn:aws:lambda:us-
west-2:123456789012:function:TrainAndBatchTransform-SeedingFunction-17RNSOTG97HPV",
      "Type": "Task",
      "Next": "Train model (XGBoost)"
    },
    "Train model (XGBoost)": {
      "Resource": "arn:aws:states:::sagemaker:createTrainingJob.sync",
      "Parameters": {
        "AlgorithmSpecification": {
          "TrainingImage": "433757028032.dkr.ecr.us-west-2.amazonaws.com/
xgboost:latest",
          "TrainingInputMode": "File"
        },
        "OutputDataConfig": {
          "S3OutputPath": "s3://trainandbatchtransform-s3bucket-1jn1le6gadwfz/models"
        },
        "StoppingCondition": {
          "MaxRuntimeInSeconds": 86400
        },
        "ResourceConfig": {
          "InstanceCount": 1,
          "InstanceType": "ml.m4.xlarge",
          "VolumeSizeInGB": 30
        },
        "RoleArn": "arn:aws:iam::123456789012:role/TrainAndBatchTransform-
SageMakerAPIExecutionRole-Y9IX3DLF6EUO",
        "InputDataConfig": [
```

```
          {
            "DataSource": {
              "S3DataSource": {
                "S3DataDistributionType": "ShardedByS3Key",
                "S3DataType": "S3Prefix",
                "S3Uri": "s3://trainandbatchtransform-s3bucket-1jn1le6gadwfz/csv/
train.csv"
              }
            },
            "ChannelName": "train",
            "ContentType": "text/csv"
          }
        ],
        "HyperParameters": {
          "objective": "reg:logistic",
          "eval_metric": "rmse",
          "num_round": "5"
        },
        "TrainingJobName.$": "$$.Execution.Name"
      },
      "Type": "Task",
      "Next": "Save Model"
    },
    "Save Model": {
      "Parameters": {
        "PrimaryContainer": {
          "Image": "433757028032.dkr.ecr.us-west-2.amazonaws.com/xgboost:latest",
          "Environment": {},
          "ModelDataUrl.$": "$.ModelArtifacts.S3ModelArtifacts"
        },
        "ExecutionRoleArn": "arn:aws:iam::123456789012:role/TrainAndBatchTransform-
SageMakerAPIExecutionRole-Y9IX3DLF6EUO",
        "ModelName.$": "$.TrainingJobName"
      },
      "Resource": "arn:aws:states:::sagemaker:createModel",
      "Type": "Task",
      "Next": "Batch transform"
    },
    "Batch transform": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sagemaker:createTransformJob.sync",
      "Parameters": {
        "ModelName.$": "$$.Execution.Name",
        "TransformInput": {
```

```
          "CompressionType": "None",
          "ContentType": "text/csv",
          "DataSource": {
            "S3DataSource": {
              "S3DataType": "S3Prefix",
              "S3Uri": "s3://trainandbatchtransform-s3bucket-1jn1le6gadwfz/csv/
test.csv"
            }
          }
        },
        "TransformOutput": {
          "S3OutputPath": "s3://trainandbatchtransform-s3bucket-1jn1le6gadwfz/output"
        },
        "TransformResources": {
          "InstanceCount": 1,
          "InstanceType": "ml.m4.xlarge"
        },
        "TransformJobName.$": "$$.Execution.Name"
      },
      "End": true
    }
  }
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Contoh IAM

Contoh kebijakan AWS Identity and Access Management (IAM) yang dihasilkan oleh proyek sampel ini mencakup hak istimewa paling sedikit yang diperlukan untuk mengeksekusi mesin negara dan sumber daya terkait. Kami merekomendasikan agar Anda hanya menyertakan izin yang diperlukan dalam kebijakan IAM Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudwatch:PutMetricData",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
```

```
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams",
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListBucket",
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

Kebijakan berikut mengizinkan fungsi Lambda untuk menempatkan bucket Amazon S3 dengan data sampel.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::trainandbatchtransform-s3bucket-1jn1le6gadwfz/*",
            "Effect": "Allow"
        }
    ]
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Setel Model Machine Learning

Proyek sampel ini menunjukkan penggunaan SageMaker untuk menyetel hiperparameter model pembelajaran mesin, dan untuk mengubah kumpulan data pengujian secara batch.

Dalam proyek ini, Step Functions menggunakan fungsi Lambda untuk menempatkan bucket Amazon S3 dengan set data uji. Ini kemudian menciptakan pekerjaan tuning hyperparameter menggunakan

integrasi SageMakerlayanan. Kemudian menggunakan fungsi Lambda untuk mengekstrak jalur data, menyimpan model tuning, mengekstrak nama model, dan kemudian menjalankan pekerjaan transformasi batch untuk melakukan inferensi di. SageMaker

Untuk informasi selengkapnya tentang SageMaker dan integrasi layanan Step Functions, lihat berikut ini:

- Menggunakan AWS Step Functions dengan layanan lain
- Kelola SageMaker dengan Step Functions

> **Note**
>
> Proyek sampel ini mungkin dikenakan biaya.
> Untuk AWS pengguna baru, tingkat penggunaan gratis tersedia. Pada tingkat ini, layanan akan gratis di bawah tingkat penggunaan tertentu. Untuk informasi selengkapnya tentang AWS biaya dan Tingkat Gratis, lihat SageMakerHarga.

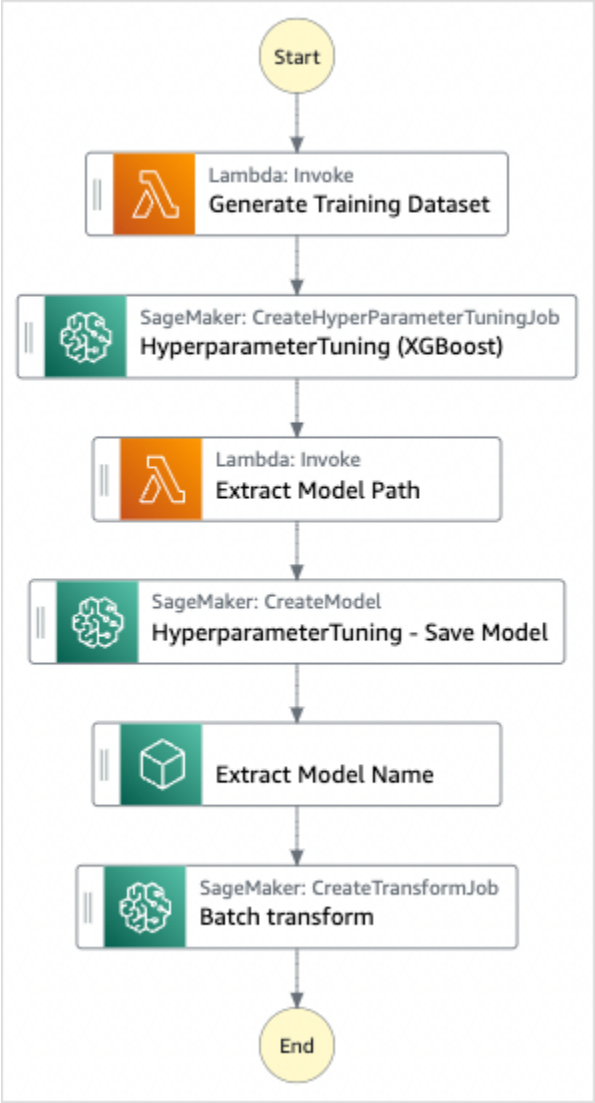
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`Tune a machine learning model`** di kotak pencarian, lalu pilih Tune model pembelajaran mesin dari hasil pencarian yang ditampilkan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

   Proyek contoh ini menyebarkan sumber daya berikut:

   - Tiga AWS Lambda fungsi
   - Bucket Amazon Simple Storage Service (Amazon S3)
   - Mesin AWS Step Functions negara
   - Peran terkait AWS Identity and Access Management (IAM)

Gambar berikut menunjukkan grafik alur kerja untuk proyek sampel model pembelajaran mesin Tune a:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

     Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon

States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

> **Important**
>
> Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

  > **Tip**
  >
  > Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

  Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

  Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

  Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

  > **Important**
  >
  > Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

1. Pada halaman mesin Negara, pilih proyek sampel Anda.

2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

    1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

        > **Note**
        >
        > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

    2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

        Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

        > **Note**
        >
        > Jika proyek demo yang Anda gunakan berisi data input eksekusi yang telah diisi sebelumnya, gunakan input tersebut untuk menjalankan mesin status.

    3. Pilih Mulai Eksekusi.
    4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

        Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Contoh Kode Mesin Status

Mesin status dalam proyek sampel ini terintegrasi dengan SageMaker dan AWS Lambda dengan meneruskan parameter langsung ke sumber daya tersebut, dan menggunakan bucket Amazon S3 untuk sumber dan output data pelatihan.

Jelajahi contoh mesin status ini untuk melihat bagaimana Step Functions mengontrol Lambda dan. SageMaker

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

```
{
    "StartAt": "Generate Training Dataset",
    "States": {
        "Generate Training Dataset": {
            "Resource": "arn:aws:lambda:us-
west-2:012345678912:function:StepFunctionsSample-SageMa-
LambdaForDataGeneration-1TF67BUE5A12U",
            "Type": "Task",
            "Next": "HyperparameterTuning (XGBoost)"
        },
        "HyperparameterTuning (XGBoost)": {
            "Resource":
 "arn:aws:states:::sagemaker:createHyperParameterTuningJob.sync",
            "Parameters": {
                "HyperParameterTuningJobName.$": "$.body.jobName",
                "HyperParameterTuningJobConfig": {
                    "Strategy": "Bayesian",
                    "HyperParameterTuningJobObjective": {
                        "Type": "Minimize",
                        "MetricName": "validation:rmse"
                    },
                    "ResourceLimits": {
                        "MaxNumberOfTrainingJobs": 2,
                        "MaxParallelTrainingJobs": 2
                    },
                    "ParameterRanges": {
                        "ContinuousParameterRanges": [{
                                "Name": "alpha",
                                "MinValue": "0",
                                "MaxValue": "1000",
                                "ScalingType": "Auto"
```

```
                            },
                            {
                                "Name": "gamma",
                                "MinValue": "0",
                                "MaxValue": "5",
                                "ScalingType": "Auto"
                            }
                        ],
                        "IntegerParameterRanges": [{
                                "Name": "max_delta_step",
                                "MinValue": "0",
                                "MaxValue": "10",
                                "ScalingType": "Auto"
                            },
                            {
                                "Name": "max_depth",
                                "MinValue": "0",
                                "MaxValue": "10",
                                "ScalingType": "Auto"
                            }
                        ]
                    }
                },
                "TrainingJobDefinition": {
                    "AlgorithmSpecification": {
                        "TrainingImage": "433757028032.dkr.ecr.us-west-2.amazonaws.com/
xgboost:latest",
                        "TrainingInputMode": "File"
                    },
                    "OutputDataConfig": {
                        "S3OutputPath": "s3://stepfunctionssample-sagemak-
bucketformodelanddata-80fblmdlcs9f/models"
                    },
                    "StoppingCondition": {
                        "MaxRuntimeInSeconds": 86400
                    },
                    "ResourceConfig": {
                        "InstanceCount": 1,
                        "InstanceType": "ml.m4.xlarge",
                        "VolumeSizeInGB": 30
                    },
                    "RoleArn": "arn:aws:iam::012345678912:role/StepFunctionsSample-
SageM-SageMakerAPIExecutionRol-1MNH1VS5CGGOG",
                    "InputDataConfig": [{
```

```
                        "DataSource": {
                            "S3DataSource": {
                                "S3DataDistributionType": "FullyReplicated",
                                "S3DataType": "S3Prefix",
                                "S3Uri": "s3://stepfunctionssample-sagemak-
bucketformodelanddata-80fblmdlcs9f/csv/train.csv"
                            }
                        },
                        "ChannelName": "train",
                        "ContentType": "text/csv"
                    },
                    {
                        "DataSource": {
                            "S3DataSource": {
                                "S3DataDistributionType": "FullyReplicated",
                                "S3DataType": "S3Prefix",
                                "S3Uri": "s3://stepfunctionssample-sagemak-
bucketformodelanddata-80fblmdlcs9f/csv/validation.csv"
                            }
                        },
                        "ChannelName": "validation",
                        "ContentType": "text/csv"
                    }],
                    "StaticHyperParameters": {
                        "precision_dtype": "float32",
                        "num_round": "2"
                    }
                }
            },
            "Type": "Task",
            "Next": "Extract Model Path"
        },
        "Extract Model Path": {
            "Resource": "arn:aws:lambda:us-
west-2:012345678912:function:StepFunctionsSample-SageM-LambdaToExtractModelPath-
V0R37CVARUS9",
            "Type": "Task",
            "Next": "HyperparameterTuning - Save Model"
        },
        "HyperparameterTuning - Save Model": {
            "Parameters": {
                "PrimaryContainer": {
                    "Image": "433757028032.dkr.ecr.us-west-2.amazonaws.com/
xgboost:latest",
```

```
                    "Environment": {},
                    "ModelDataUrl.$": "$.body.modelDataUrl"
                },
                "ExecutionRoleArn": "arn:aws:iam::012345678912:role/
StepFunctionsSample-SageM-SageMakerAPIExecutionRol-1MNH1VS5CGGOG",
                "ModelName.$": "$.body.bestTrainingJobName"
            },
            "Resource": "arn:aws:states:::sagemaker:createModel",
            "Type": "Task",
            "Next": "Extract Model Name"
        },
        "Extract Model Name": {
            "Resource": "arn:aws:lambda:us-
west-2:012345678912:function:StepFunctionsSample-SageM-
LambdaToExtractModelName-8FUOB30SM5EM",
            "Type": "Task",
            "Next": "Batch transform"
        },
        "Batch transform": {
            "Type": "Task",
            "Resource": "arn:aws:states:::sagemaker:createTransformJob.sync",
            "Parameters": {
                "ModelName.$": "$.body.jobName",
                "TransformInput": {
                    "CompressionType": "None",
                    "ContentType": "text/csv",
                    "DataSource": {
                        "S3DataSource": {
                            "S3DataType": "S3Prefix",
                            "S3Uri": "s3://stepfunctionssample-sagemak-
bucketformodelanddata-80fblmdlcs9f/csv/test.csv"
                        }
                    }
                },
                "TransformOutput": {
                    "S3OutputPath": "s3://stepfunctionssample-sagemak-
bucketformodelanddata-80fblmdlcs9f/output"
                },
                "TransformResources": {
                    "InstanceCount": 1,
                    "InstanceType": "ml.m4.xlarge"
                },
                "TransformJobName.$": "$.body.jobName"
            },
```

```
            "End": true
        }
    }
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan layanan AWS lain, lihat Kebijakan IAM untuk layanan terintegrasi.

Contoh IAM

Contoh kebijakan AWS Identity and Access Management (IAM) yang dihasilkan oleh proyek sampel ini mencakup hak istimewa paling sedikit yang diperlukan untuk mengeksekusi mesin negara dan sumber daya terkait. Kami merekomendasikan agar Anda hanya menyertakan izin yang diperlukan dalam kebijakan IAM Anda.

Kebijakan IAM berikut dilampirkan ke mesin negara, dan memungkinkan eksekusi mesin negara untuk mengakses sumber daya yang diperlukan SageMaker, Lambda, dan Amazon S3.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sagemaker:CreateHyperParameterTuningJob",
                "sagemaker:DescribeHyperParameterTuningJob",
                "sagemaker:StopHyperParameterTuningJob",
                "sagemaker:ListTags",
                "sagemaker:CreateModel",
                "sagemaker:CreateTransformJob",
                "iam:PassRole"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-west-2:012345678912:function:StepFunctionsSample-
SageMa-LambdaForDataGeneration-1TF67BUE5A12U",
```

```
                "arn:aws:lambda:us-west-2:012345678912:function:StepFunctionsSample-
SageM-LambdaToExtractModelPath-V0R37CVARUS9",
                "arn:aws:lambda:us-west-2:012345678912:function:StepFunctionsSample-
SageM-LambdaToExtractModelName-8FUOB30SM5EM"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:*:*:rule/
StepFunctionsGetEventsForSageMakerTrainingJobsRule",
                "arn:aws:events:*:*:rule/
StepFunctionsGetEventsForSageMakerTransformJobsRule",
                "arn:aws:events:*:*:rule/
StepFunctionsGetEventsForSageMakerTuningJobsRule"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Kebijakan IAM berikut direferensikan di bidang `TrainingJobDefinition` dan `HyperparameterTuning` status `HyperparameterTuning`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudwatch:PutMetricData",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams",
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
```

```
                "sagemaker:DescribeHyperParameterTuningJob",
                "sagemaker:StopHyperParameterTuningJob",
                "sagemaker:ListTags"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::stepfunctionssample-sagemak-
bucketformodelanddata-80fblmdlcs9f/*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::stepfunctionssample-sagemak-
bucketformodelanddata-80fblmdlcs9f",
            "Effect": "Allow"
        }
    ]
}
```

Kebijakan IAM berikut mengizinkan fungsi Lambda untuk menempatkan bucket Amazon S3 dengan data sampel.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::stepfunctionssample-sagemak-
bucketformodelanddata-80fblmdlcs9f/*",
            "Effect": "Allow"
        }
    ]
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan layanan AWS lain, lihat Kebijakan IAM untuk layanan terintegrasi.

Proses Pesan Bervolume Tinggi dari Amazon SQS (Alur Kerja Express)

Proyek contoh ini menunjukkan cara menggunakan Alur Kerja AWS Step Functions Ekspres untuk memproses pesan atau data dari sumber peristiwa bervolume tinggi, seperti Amazon Simple Queue Service (Amazon SQS). Karena Alur Kerja Ekspres dapat dimulai dengan kecepatan yang sangat tinggi, Alur kerja tersebut ideal untuk pemrosesan peristiwa atau beban kerja streaming data volume tinggi.

Berikut adalah dua metode yang umum digunakan untuk menjalankan mesin status Anda dari sumber peristiwa:

- Konfigurasikan aturan Amazon CloudWatch Events untuk memulai eksekusi mesin status setiap kali sumber peristiwa memancarkan peristiwa. Untuk informasi selengkapnya, lihat Membuat Aturan CloudWatch Peristiwa yang Memicu Peristiwa.
- Petakan sumber peristiwa ke fungsi Lambda, dan tulis kode fungsi untuk mengeksekusi mesin status Anda. AWS Lambda Fungsi ini dipanggil setiap kali sumber acara Anda memancarkan peristiwa, pada gilirannya memulai eksekusi mesin status. Untuk informasi selengkapnya, lihat Menggunakan AWS Lambda dengan Amazon SQS.

Proyek sampel ini menggunakan metode kedua untuk memulai eksekusi setiap kali antrean Amazon SQS mengirim pesan. Anda dapat menggunakan konfigurasi serupa untuk memicu eksekusi Alur Kerja Express dari sumber peristiwa lain, seperti Amazon Simple Storage Service (Amazon S3), Amazon DynamoDB, dan Amazon Kinesis.

Untuk informasi selengkapnya tentang Alur Kerja Express dan integrasi layanan Step Functions, lihat di bawah ini:

- Alur Kerja Standar vs Ekspres
- Menggunakan AWS Step Functions dengan layanan lain
- Kuota

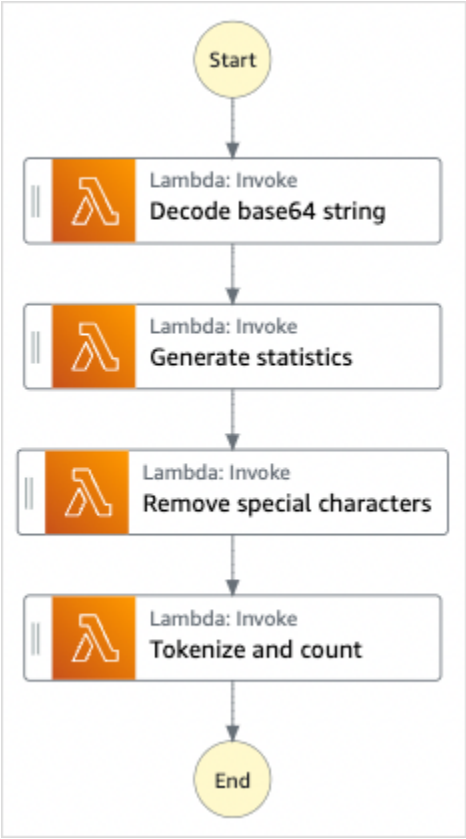
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`Process high-volume messages from SQS`** di kotak pencarian, lalu pilih Proses pesan volume tinggi dari SQS dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

   Proyek contoh ini menyebarkan sumber daya berikut:

   - Empat fungsi Lambda
   - Antrean Amazon SQS
   - Mesin AWS Step Functions negara
   - Peran terkait AWS Identity and Access Management (IAM)

   Gambar berikut menunjukkan grafik alur kerja untuk Proses pesan volume tinggi dari proyek sampel SQS:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu dari cara berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

     Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

> **Important**
>
> Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

  > **Tip**
  >
  > Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

  Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

  Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

  Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

  > **Important**
  >
  > Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Memicu eksekusi mesin negara

1. Buka konsol Amazon SQS.
2. Pilih antrean yang dibuat oleh proyek sampel.

   Namanya akan sama dengan Example-SQSQueue-wJalrXUtnFEMI.

3. Di daftar Tindakan antrean, pilih Kirim pesan.
4. Gunakan tombol salin untuk menyalin pesan berikut, dan pada jendela Kirim pesan, masukkan pesan tersebut, dan pilih Kirim pesan.

> **Note**
>
> Dalam pesan sampel ini, baris `input:` telah diformat dengan jeda baris agar sesuai dengan halaman. Gunakan tombol salin atau memastikan bahwa pesan dimasukkan sebagai satu baris tanpa jeda.

```
{
      "input":
 "QW5kIGxpa2UgdGhlIGJhc2VsZXNzIGZhYnJpYyBvZiB0aGlzIHZpc2lvbiwgVGhlIGNsb3VkLWNhcHBlZCB0b3dlc

  91cyBwYWxhY2VzLCBUaGUgc29sZW1uIHRlbXBsZXMsIHRoZSBncmVhdCBnbG9iZSBpdHNlbGbigJQgWWVhLCBhbGw

  ZXJpdOKAlHNoYWxsIGRpc3NvbHZlLCBBbmQgbGlrZSB0aGlzIGluc3Vic3RhbnRpYWwgcGFnZWFudCBmYWRlZCwgT

  FjayBiZWhpbmQuIFdlIGFyZSBzdWNoIHN0dWZmIEFzIGRyZWFtcyBhcmUgbWFkZSBvbiwgYW5kIG91ciBsaXR0bGU

  ZGVkIHdpdGggYSBzbGVlcC4gU2lyLCBJIGFtIHZleGVkLiBCZWFyIHdpdGggbXkgd2Vha25lc3MuIE15IG9sZCBic

  xlZC4gQmUgbm90IGRpc3R1cmJlZCB3aXRoIG15IGluZmlybWl0eS4gSWYgeW91IGJlIHBsZWFzZWQsIHJldGlyZSB

  QW5kIHRoZXJlIHJlcG9zZS4gQSB0dXJuIG9yIHR3byBJ4oCZbGwgd2FsayBUbyBzdGlsbCBteSBzdGlsbCBteSBiZWF0aW5nIG1pb

}
```

5. Pilih Tutup.
6. Buka Konsol Step Functions.
7. Buka grup CloudWatch log Amazon Logs Anda dan periksa log. Nama grup log akan terlihat seperti contoh- ExpressLogGroup -WJAlrxutnFemi.

Contoh Kode Fungsi Lambda

Berikut ini adalah kode fungsi Lambda yang menunjukkan bagaimana fungsi Lambda yang memulai memulai eksekusi mesin status menggunakan SDK. AWS

```
import boto3
```

```
def lambda_handler(event, context):
    message_body = event['Records'][0]['body']
    client = boto3.client('stepfunctions')
    response = client.start_execution(
        stateMachineArn='${ExpressStateMachineArn}',
        input=message_body
    )
```

Contoh Kode Mesin Status

Alur Kerja Express dalam proyek sampel ini terdiri dari satu set fungsi Lambda untuk pemrosesan teks.

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

```
{
  "Comment": "An example of using Express workflows to run text processing for each
 message sent from an SQS queue.",
  "StartAt": "Decode base64 string",
  "States": {
    "Decode base64 string": {
      "Type": "Task",
      "Resource": "arn:<PARTITION>:states:::lambda:invoke",
      "OutputPath": "$.Payload",
      "Parameters": {
        "FunctionName": "<BASE64_DECODER_LAMBDA_FUNCTION_NAME>",
        "Payload.$": "$"
      },
      "Next": "Generate statistics"
    },
    "Generate statistics": {
      "Type": "Task",
      "Resource": "arn:<PARTITION>:states:::lambda:invoke",
      "OutputPath": "$.Payload",
      "Parameters": {
        "FunctionName": "<TEXT_STATS_GENERATING_LAMBDA_FUNCTION_NAME>",
        "Payload.$": "$"
      },
      "Next": "Remove special characters"
    },
    "Remove special characters": {
```

```
      "Type": "Task",
      "Resource": "arn:<PARTITION>:states:::lambda:invoke",
      "OutputPath": "$.Payload",
      "Parameters": {
        "FunctionName": "<STRING_CLEANING_LAMBDA_FUNCTION_NAME>",
        "Payload.$": "$"
      },
      "Next": "Tokenize and count"
    },
    "Tokenize and count": {
      "Type": "Task",
      "Resource": "arn:<PARTITION>:states:::lambda:invoke",
      "OutputPath": "$.Payload",
      "Parameters": {
        "FunctionName": "<TOKENIZING_AND_WORD_COUNTING_LAMBDA_FUNCTION_NAME>",
        "Payload.$": "$"
      },
      "End": true
    }
  }
}
```

Contoh IAM

Contoh kebijakan AWS Identity and Access Management (IAM) yang dihasilkan oleh proyek sampel ini mencakup hak istimewa paling sedikit yang diperlukan untuk mengeksekusi mesin negara dan sumber daya terkait. Kami merekomendasikan agar Anda hanya menyertakan izin yang diperlukan dalam kebijakan IAM Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:123456789012:function:example-
Base64DecodeLambda-wJalrXUtnFEMI",
                "arn:aws:lambda:us-east-1:123456789012:function:example-
StringCleanerLambda-je7MtGbClwBF",
                "arn:aws:lambda:us-east-1:123456789012:function:example-
TokenizerCounterLambda-wJalrXUtnFEMI",
```

```
                "arn:aws:lambda:us-east-1:123456789012:function:example-
GenerateStatsLambda-je7MtGbClwBF"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Kebijakan berikut memastikan bahwa ada izin yang cukup untuk CloudWatch Log.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries",
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Contoh Checkpointing selektif (Alur kerja Express)

Proyek sampel ini menunjukkan cara menggabungkan Alur Kerja Standar dan Ekspres dengan menjalankan alur kerja perdagangan elektronik tiruan yang melakukan checkpointing selektif. Men-deploy proyek sampel membuat mesin status alur kerja Standar, mesin status ekspres Alur Kerja

Express nest, fungsi AWS Lambda , Antrean Amazon Simple Queue Service (Amazon SQS), dan topik Amazon Simple Notification Service (Amazon SNS).

Untuk informasi selengkapnya tentang alur Kerja Express, alur Kerja nest, dan integrasi layanan Step Functions, lihat di bawah ini:

- Alur Kerja Standar vs Ekspres
- Mulai Eksekusi Alur Kerja dari Status Tugas.
- Menggunakan AWS Step Functions dengan layanan lain

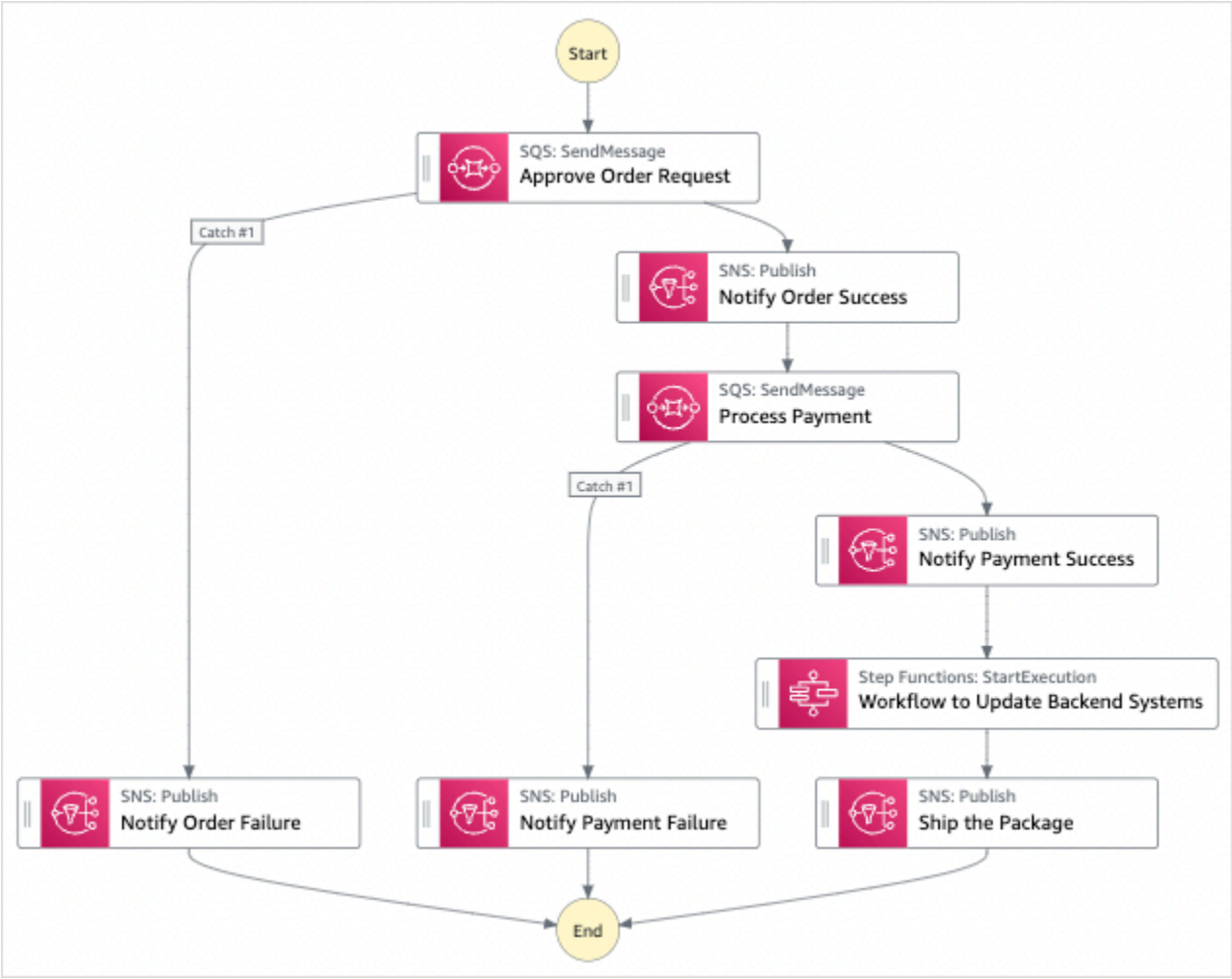
Langkah 1: Buat Mesin Negara dan Sumber Daya Penyediaan

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`Selective checkpointing example`** di kotak pencarian, lalu pilih Contoh checkpointing selektif dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

   Proyek contoh ini menyebarkan sumber daya berikut:

   - Sebuah AWS Lambda fungsi
   - Antrean Amazon SQS
   - Topik Amazon SNS
   - Mesin AWS Step Functions negara tipe Standar
   - Mesin status Step Functions tipe Express
   - Peran terkait AWS Identity and Access Management (IAM)

   Gambar berikut menunjukkan grafik alur kerja untuk proyek contoh contoh checkpointing Selective:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

     Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

> **Important**
>
> Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

  > **Tip**
  >
  > Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

  Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

  Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

  Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

  > **Important**
  >
  > Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Setelah sumber daya dari proyek sampel di-deploy lakukan hal berikut.

Langkah 2: Jalankan mesin negara

1. Pada halaman mesin Negara, pilih proyek sampel Anda.
2. Pada halaman proyek sampel, pilih Mulai eksekusi.

3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

    1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

        > **Note**
        >
        > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

    2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

        Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

        > **Note**
        >
        > Jika proyek demo yang Anda gunakan berisi data input eksekusi yang telah diisi sebelumnya, gunakan input tersebut untuk menjalankan mesin status.

    3. Pilih Mulai Eksekusi.

    4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

        Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

4. Buka grup CloudWatch log Log Anda dan periksa log. Nama grup log akan terlihat seperti contoh- ExpressLogGroup -WJAlrxutnFemi.

Contoh Kode Mesin Status untuk Induk (Standar Alur Kerja)

Mesin status dalam proyek sampel ini terintegrasi dengan Amazon SQS, Amazon SNS, dan Alur Kerja Express Step Functions.

Jelajahi melalui contoh mesin status ini untuk melihat cara Step Functions memproses masukan dari Amazon SQS dan Amazon SNS, lalu menggunakan mesin status Alur Kerja Express nest untuk memperbarui sistem backend.

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

```
{
    "Comment": "An example of combining standard and express workflows to run a mock e-
commerce workflow that does selective checkpointing.",
    "StartAt": "Approve Order Request",
    "States": {
        "Approve Order Request": {
            "Type": "Task",
            "Resource": "arn:<PARTITION>:states:::sqs:sendMessage.waitForTaskToken",
            "Parameters": {
                "QueueUrl": "<SQS_QUEUE_URL>",
                "MessageBody": {
                    "MessageTitle": "Order Request received. Pausing workflow to wait
 for manual approval. ",
                    "TaskToken.$": "$$.Task.Token"
                }
            },
            "Next": "Notify Order Success",
            "Catch": [
                {
                    "ErrorEquals": [
                        "States.ALL"
                    ],
                    "Next": "Notify Order Failure"
                }
            ]
        },
        "Notify Order Success": {
            "Type": "Task",
            "Resource": "arn:<PARTITION>:states:::sns:publish",
            "Parameters": {
                "Message": "Order has been approved. Resuming workflow.",
```

```
                "TopicArn": "<SNS_ARN>"
            },
            "Next": "Process Payment"
        },
        "Notify Order Failure": {
            "Type": "Task",
            "Resource": "arn:<PARTITION>:states:::sns:publish",
            "Parameters": {
                "Message": "Order not approved. Order failed.",
                "TopicArn": "<SNS_ARN>"
            },
            "End": true
        },
        "Process Payment": {
            "Type": "Task",
            "Resource": "arn:<PARTITION>:states:::sqs:sendMessage.waitForTaskToken",
            "Parameters": {
                "QueueUrl": "<SQS_QUEUE_URL>",
                "MessageBody": {
                    "MessageTitle": "Payment sent to third-party for processing.
Pausing workflow to wait for response.",
                    "TaskToken.$": "$$.Task.Token"
                }
            },
            "Next": "Notify Payment Success",
            "Catch": [
                {
                    "ErrorEquals": [
                        "States.ALL"
                    ],
                    "Next": "Notify Payment Failure"
                }
            ]
        },
        "Notify Payment Success": {
            "Type": "Task",
            "Resource": "arn:<PARTITION>:states:::sns:publish",
            "Parameters": {
                "Message": "Payment processing succeeded. Resuming workflow.",
                "TopicArn": "<SNS_ARN>"
            },
            "Next": "Workflow to Update Backend Systems"
        },
        "Notify Payment Failure": {
```

```
            "Type": "Task",
            "Resource": "arn:<PARTITION>:states:::sns:publish",
            "Parameters": {
                "Message": "Payment processing failed.",
                "TopicArn": "<SNS_ARN>"
            },
            "End": true
        },
        "Workflow to Update Backend Systems": {
            "Comment": "Starting an execution of an Express workflow to handle backend
 updates. Express workflows are fast and cost-effective for steps where checkpointing
 isn't required.",
            "Type": "Task",
            "Resource": "arn:<PARTITION>:states:::states:startExecution.sync",
            "Parameters": {
                "StateMachineArn": "<UPDATE_DATABASE_EXPRESS_STATE_MACHINE_ARN>",
                "Input": {
                    "AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID.$": "$$.Execution.Id"
                }
            },
            "Next": "Ship the Package"
        },
        "Ship the Package": {
            "Type": "Task",
            "Resource": "arn:<PARTITION>:states:::sns:publish",
            "Parameters": {
                "Message": "Order and payment received, database is updated and the
 package is ready to ship.",
                "TopicArn": "<SNS_ARN>"
            },
            "End": true
        }
    }
}
```

Contoh IAM role untuk Mesin Status Induk

Contoh kebijakan AWS Identity and Access Management (IAM) yang dihasilkan oleh proyek sampel ini mencakup hak istimewa paling sedikit yang diperlukan untuk mengeksekusi mesin negara dan sumber daya terkait. Kami merekomendasikan agar Anda hanya menyertakan izin yang diperlukan dalam kebijakan IAM Anda.

Kebijakan Amazon SNS:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": "arn:aws:sns:us-east-1:123456789012:Checkpoint-SNSTopic-
wJalrXUtnFEMI",
            "Effect": "Allow"
        }
    ]
}
```

Kebijakan Amazon SQS:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sqs:SendMessage"
            ],
            "Resource": "arn:aws:sqs:us-east-1:123456789012:Checkpoint-SQSQueue-
je7MtGbClwBF",
            "Effect": "Allow"
        }
    ]
}
```

Kebijakan eksekusi status:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "states:StartExecution",
                "states:DescribeExecution",
                "states:StopExecution"
            ],
            "Resource": "*",
```

```
            "Effect": "Allow"
        },
        {
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": "arn:aws:events:us-east-1:123456789012:rule/
StepFunctionsGetEventsForStepFunctionsExecutionRule",
            "Effect": "Allow"
        }
    ]
}
```

Contoh Kode Mesin Status untuk Mesin Status Nest (Alur Kerja Express)

Mesin status dalam proyek sampel ini memperbarui informasi backend ketika dipanggil oleh mesin status induk.

Jelajahi melalui contoh mesin status ini untuk melihat cara Step Functions memperbarui komponen yang berbeda dari tiruan sistem backend perdagangan elektronik.

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

```
{
  "StartAt": "Update Order History",
  "States": {
    "Update Order History": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "Parameters": {
        "FunctionName": "Checkpoint-UpdateDatabaseLambdaFunction-wJalrXUtnFEMI",
        "Payload": {
          "Message": "Update order history."
        }
      },
      "Next": "Update Data Warehouse"
    },
    "Update Data Warehouse": {
      "Type" : "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "Parameters": {
        "FunctionName": "Checkpoint-UpdateDatabaseLambdaFunction-wJalrXUtnFEMI",
        "Payload": {
          "Message": "Update data warehouse."
        }
      },
```

```
      "Next": "Update Customer Profile"
    },
    "Update Customer Profile": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "Parameters": {
        "FunctionName": "Checkpoint-UpdateDatabaseLambdaFunction-wJalrXUtnFEMI",
        "Payload": {
          "Message": "Update customer profile."
        }
      },
      "Next": "Update Inventory"
    },
    "Update Inventory": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "Parameters": {
        "FunctionName": "Checkpoint-UpdateDatabaseLambdaFunction-wJalrXUtnFEMI",
        "Payload": {
          "Message": "Update inventory."
        }
      },
      "End": true
    }
  }
}
```

Contoh IAM role untuk Mesin Status Anak

Contoh kebijakan AWS Identity and Access Management (IAM) yang dihasilkan oleh proyek sampel ini mencakup hak istimewa paling sedikit yang diperlukan untuk mengeksekusi mesin negara dan sumber daya terkait. Kami merekomendasikan agar Anda hanya menyertakan izin yang diperlukan dalam kebijakan IAM Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
```

```
                "arn:aws:lambda:us-east-1:123456789012:function:Example-
UpdateDatabaseLambdaFunction-wJalrXUtnFEMI"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Kebijakan berikut memastikan bahwa ada izin yang cukup untuk CloudWatch Log.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries",
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan layanan AWS lain, lihat Kebijakan IAM untuk layanan terintegrasi.

Membangun AWS CodeBuild Proyek (CodeBuild, Amazon SNS)

Proyek contoh ini mendemonstrasikan cara menggunakan AWS Step Functions untuk membangun AWS CodeBuild proyek, menjalankan pengujian, dan kemudian mengirim notifikasi Amazon SNS.

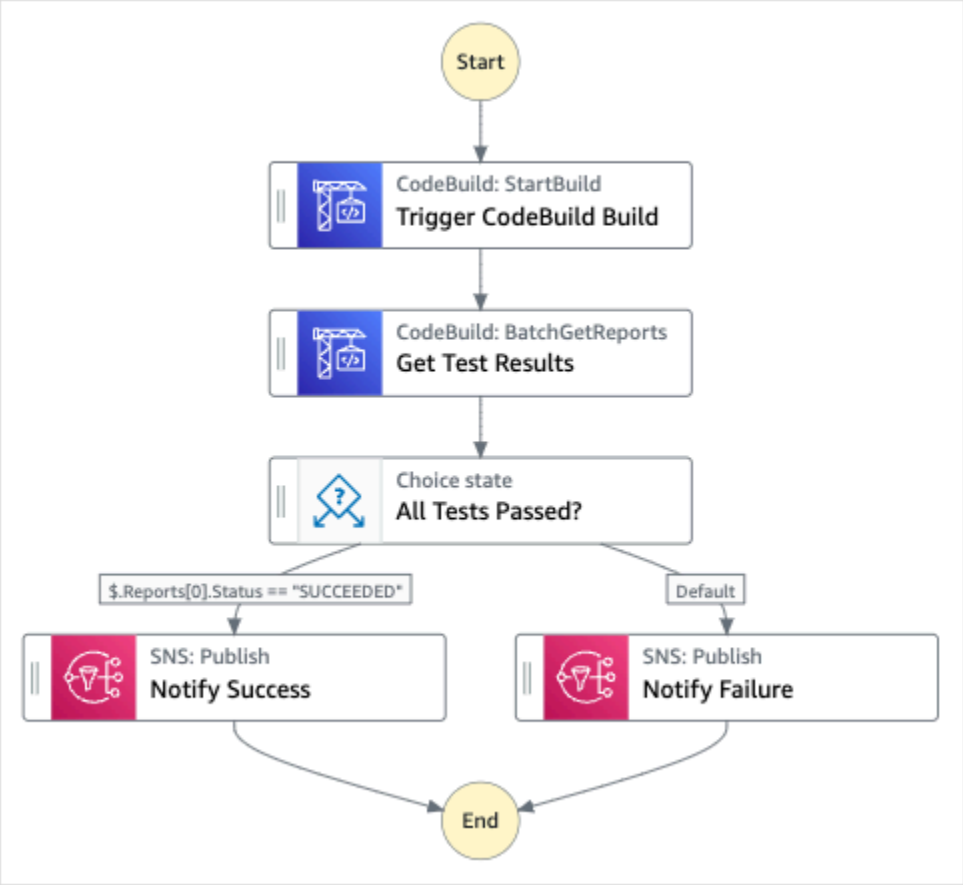
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`Start a CodeBuild build`** di kotak pencarian, lalu pilih Mulai CodeBuild build dari hasil penelusuran yang ditampilkan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

   Proyek contoh ini menyebarkan sumber daya berikut:

   - Sebuah AWS CodeBuild bangunan
   - Topik Amazon SNS
   - Mesin AWS Step Functions negara
   - Peran terkait AWS Identity and Access Management (IAM)

   Gambar berikut menunjukkan grafik alur kerja untuk proyek sampel Mulai CodeBuild membangun:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

     Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

> **Important**
>
> Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

> **Tip**
>
> Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

> **Important**
>
> Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

1. Pada halaman mesin Negara, pilih proyek sampel Anda.
2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

   > **Note**
   >
   > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

   Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

   > **Note**
   >
   > Jika proyek demo yang Anda gunakan berisi data input eksekusi yang telah diisi sebelumnya, gunakan input tersebut untuk menjalankan mesin status.

3. Pilih Mulai Eksekusi.
4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

   Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Contoh Kode Mesin Status

Mesin status dalam proyek sampel ini terintegrasi dengan CodeBuild dan Amazon SNS.

Jelajahi contoh mesin status ini untuk melihat bagaimana Step Functions menggunakan mesin status untuk membangun CodeBuild proyek, lalu mengirimkan topik Amazon SNS dengan pesan tentang apakah pekerjaan berhasil atau gagal.

Untuk informasi selengkapnya tentang cara Step Functions dapat mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

```
{
  "Comment": "An example of using CodeBuild to run tests, get test results and send a
 notification.",
  "StartAt": "Trigger CodeBuild Build",
  "States": {
    "Trigger CodeBuild Build": {
      "Type": "Task",
      "Resource": "arn:aws:states:::codebuild:startBuild.sync",
      "Parameters": {
        "ProjectName": "CodeBuildProject-Dtw1jBhEYGDf"
      },
      "Next": "Get Test Results"
    },
    "Get Test Results": {
      "Type": "Task",
      "Resource": "arn:aws:states:::codebuild:batchGetReports",
      "Parameters": {
        "ReportArns.$": "$.Build.ReportArns"
      },
      "Next": "All Tests Passed?"
    },
    "All Tests Passed?": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.Reports[0].Status",
          "StringEquals": "SUCCEEDED",
          "Next": "Notify Success"
        }
      ],
      "Default": "Notify Failure"
    },
    "Notify Success": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
```

```
        "Message": "CodeBuild build tests succeeded",
        "TopicArn": "arn:aws:sns:sa-east-1:123456789012:StepFunctionsSample-
CodeBuildExecution3da9ead6-bc1f-4441-99ac-591c140019c4-SNSTopic-EVYLVNGW85JP"
      },
      "End": true
    },
    "Notify Failure": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
        "Message": "CodeBuild build tests failed",
        "TopicArn": "arn:aws:sns:sa-east-1:123456789012:StepFunctionsSample-
CodeBuildExecution3da9ead6-bc1f-4441-99ac-591c140019c4-SNSTopic-EVYLVNGW85JP"
      },
      "End": true
    }
  }
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan layanan AWS lain, lihat Kebijakan IAM untuk layanan terintegrasi.

Praproses data dan latih model machine learning

Proyek sampel ini menunjukkan bagaimana menggunakan SageMaker dan AWS Step Functions memproses data sebelumnya dan melatih model pembelajaran mesin.

Dalam proyek ini, Step Functions menggunakan fungsi Lambda untuk menempatkan bucket Amazon S3 dengan set data uji dan skrip Python untuk memproses data. Kemudian melatih model pembelajaran mesin dan melakukan transformasi batch, menggunakan integrasi SageMaker layanan.

Untuk informasi selengkapnya tentang SageMaker dan integrasi layanan Step Functions, lihat berikut ini:

- Menggunakan AWS Step Functions dengan layanan lain
- Kelola SageMaker dengan Step Functions

> **Note**
>
> Proyek sampel ini mungkin dikenakan biaya.

Untuk AWS pengguna baru, tingkat penggunaan gratis tersedia. Pada tingkat ini, layanan akan gratis di bawah tingkat penggunaan tertentu. Untuk informasi selengkapnya tentang AWS biaya dan Tingkat Gratis, lihat SageMaker Harga.

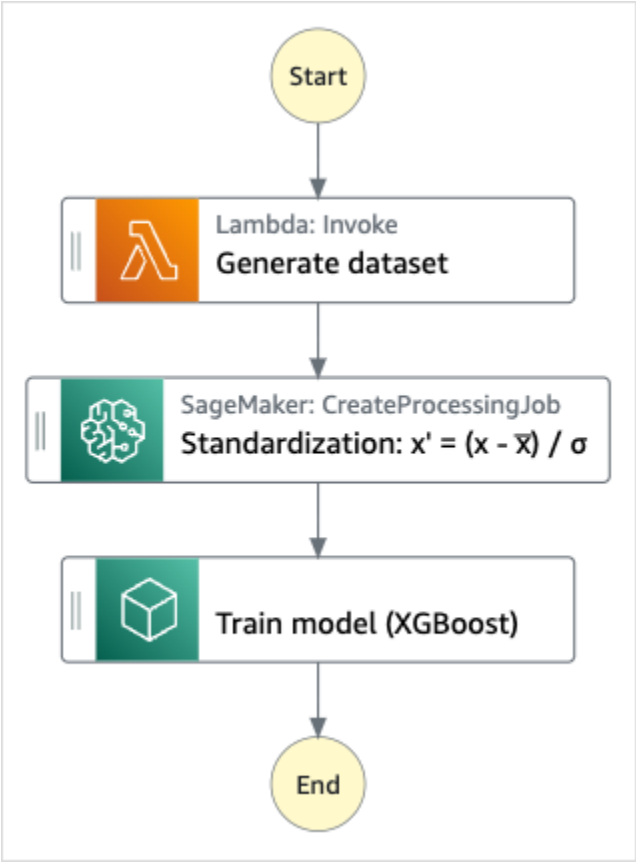
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`Preprocess data and train a machine learning model`** di kotak pencarian, lalu pilih Data pra-proses dan latih model pembelajaran mesin dari hasil penelusuran yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

   Proyek contoh ini menyebarkan sumber daya berikut:

   - Sebuah AWS Lambda fungsi
   - Bucket Amazon S3
   - Mesin AWS Step Functions negara
   - Peran terkait AWS Identity and Access Management (IAM)

   Gambar berikut menunjukkan grafik alur kerja untuk data Preprocess dan melatih proyek contoh model pembelajaran mesin:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

     Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

> **Important**
>
> Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

> **Tip**
>
> Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

> **Important**
>
> Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

1. Pada halaman mesin Negara, pilih proyek sampel Anda.
2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

   > **Note**
   >
   > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

   Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

   > **Note**
   >
   > Jika proyek demo yang Anda gunakan berisi data input eksekusi yang telah diisi sebelumnya, gunakan input tersebut untuk menjalankan mesin status.

3. Pilih Mulai Eksekusi.
4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

   Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Contoh Kode Mesin Status

Mesin status dalam proyek sampel ini terintegrasi dengan SageMaker dan AWS Lambda dengan meneruskan parameter langsung ke sumber daya tersebut, dan menggunakan bucket Amazon S3 untuk sumber dan output data pelatihan.

Jelajahi contoh mesin status ini untuk melihat bagaimana Step Functions mengontrol Lambda dan. SageMaker

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

```
{
  "StartAt": "Generate dataset",
  "States": {
    "Generate dataset": {
      "Resource": "arn:aws:lambda:sa-east-1:1234567890:function:FeatureTransform-
LambaForDataGeneration-17M8LX7IO9LUW",
      "Type": "Task",
      "Next": "Standardization: x' = (x - x̄) / σ"
    },
    "Standardization: x' = (x - x̄) / σ": {
      "Resource": "arn:aws:states:::sagemaker:createProcessingJob.sync",
      "Parameters": {
        "ProcessingResources": {
          "ClusterConfig": {
            "InstanceCount": 1,
            "InstanceType": "ml.m5.xlarge",
            "VolumeSizeInGB": 10
          }
        },
        "ProcessingInputs": [
            {
                "InputName": "input-1",
                "S3Input": {
                    "S3Uri": "s3://featuretransform-bucketforcodeanddata-1jn1le6gadwfz/
input/raw.csv",
                    "LocalPath": "/opt/ml/processing/input",
                    "S3DataType": "S3Prefix",
                    "S3InputMode": "File",
                    "S3DataDistributionType": "FullyReplicated",
                    "S3CompressionType": "None"
                }
            },
            {
                "InputName": "code",
                "S3Input": {
                    "S3Uri": "s3://featuretransform-bucketforcodeanddata-1jn1le6gadwfz/
code/transform.py",
```

```
                    "LocalPath": "/opt/ml/processing/input/code",
                    "S3DataType": "S3Prefix",
                    "S3InputMode": "File",
                    "S3DataDistributionType": "FullyReplicated",
                    "S3CompressionType": "None"
                }
            }
        ],
        "ProcessingOutputConfig": {
            "Outputs": [
                {
                    "OutputName": "train_data",
                    "S3Output": {
                        "S3Uri": "s3://featuretransform-
bucketforcodeanddata-1jn1le6gadwfz/train",
                        "LocalPath": "/opt/ml/processing/output/train",
                        "S3UploadMode": "EndOfJob"
                    }
                }
            ]
        },
        "AppSpecification": {
            "ImageUri": "737474898029.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-scikit-
learn:0.20.0-cpu-py3",
            "ContainerEntrypoint": [
                "python3",
                "/opt/ml/processing/input/code/transform.py"
            ]
        },
        "StoppingCondition": {
            "MaxRuntimeInSeconds": 300
        },
        "RoleArn": "arn:aws:iam::1234567890:role/SageMakerAPIExecutionRole-
AIDACKCEVSQ6C2EXAMPLE",
        "ProcessingJobName.$": "$$.Execution.Name"
      },
      "Type": "Task",
      "Next": "Train model (XGBoost)"
    },
    "Train model (XGBoost)": {
      "Resource": "arn:aws:states:::sagemaker:createTrainingJob.sync",
      "Parameters": {
        "AlgorithmSpecification": {
```

```
          "TrainingImage": "855470959533.dkr.ecr.sa-east-1.amazonaws.com/
xgboost:latest",
          "TrainingInputMode": "File"
        },
        "OutputDataConfig": {
          "S3OutputPath": "s3://featuretransform-bucketforcodeanddata-1jn1le6gadwfz/
models"
        },
        "StoppingCondition": {
          "MaxRuntimeInSeconds": 86400
        },
        "ResourceConfig": {
          "InstanceCount": 1,
          "InstanceType": "ml.m5.xlarge",
          "VolumeSizeInGB": 30
        },
        "RoleArn": "arn:aws:iam::1234567890:role/SageMakerAPIExecutionRole-
AIDACKCEVSQ6C2EXAMPLE",
        "InputDataConfig": [
          {
            "DataSource": {
              "S3DataSource": {
                "S3DataDistributionType": "ShardedByS3Key",
                "S3DataType": "S3Prefix",
                "S3Uri": "s3://featuretransform-bucketforcodeanddata-1jn1le6gadwfz"
              }
            },
            "ChannelName": "train",
            "ContentType": "text/csv"
          }
        ],
        "HyperParameters": {
          "objective": "reg:logistic",
          "eval_metric": "rmse",
          "num_round": "5"
        },
        "TrainingJobName.$": "$$.Execution.Name"
      },
      "Type": "Task",
      "End": true
    }
  }
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Contoh IAM

Contoh kebijakan AWS Identity and Access Management (IAM) yang dihasilkan oleh proyek sampel ini mencakup hak istimewa paling sedikit yang diperlukan untuk mengeksekusi mesin negara dan sumber daya terkait. Kami merekomendasikan agar Anda hanya menyertakan izin yang diperlukan dalam kebijakan IAM Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudwatch:PutMetricData",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams",
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListBucket",
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

Kebijakan berikut mengizinkan fungsi Lambda untuk menempatkan bucket Amazon S3 dengan data sampel.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
```

```
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::featuretransform-
bucketforcodeanddata-1jn1le6gadwfz/*",
            "Effect": "Allow"
        }
    ]
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Contoh orkestrasi Lambda

Proyek sampel ini menunjukkan cara mengintegrasikan AWS Lambda fungsi dalam mesin status Step Functions.

Dalam proyek ini, Step Functions menggunakan fungsi Lambda untuk memeriksa harga saham dan menentukan rekomendasi perdagangan beli atau jual. Pengguna kemudian memberikan rekomendasi ini dan dapat memilih apakah akan membeli atau menjual saham. Hasil perdagangan dikembalikan menggunakan topik SNS.

Untuk informasi selengkapnya tentang integrasi layanan Step Functions, lihat hal berikut:

- Menggunakan AWS Step Functions dengan layanan lain
- Kebijakan IAM untuk:
  - Kebijakan IAM untuk AWS Lambda
  - Kebijakan IAM untuk Amazon SQS
  - Kebijakan IAM untuk Amazon SNS

> **Note**
>
> Proyek sampel ini mungkin dikenakan biaya.
> Untuk AWS pengguna baru, tingkat penggunaan gratis tersedia. Pada tingkat ini, layanan akan gratis di bawah tingkat penggunaan tertentu. Untuk informasi selengkapnya tentang AWS biaya dan tingkat gratis, lihat Harga.

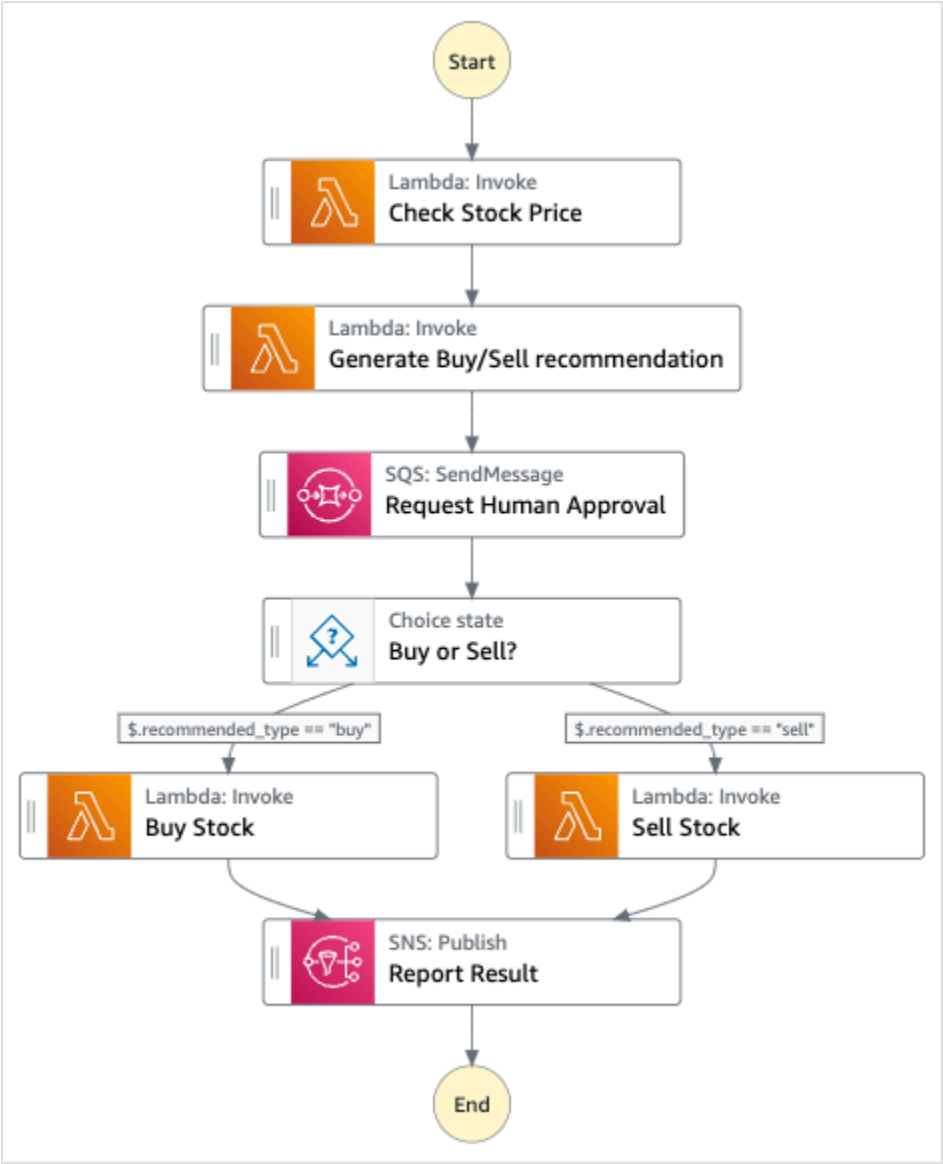
Langkah 1: Buat mesin negara dan sumber daya penyediaan

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`Orchestrate Lambda functions`** di kotak pencarian, lalu pilih Orkestrasi Lambda fungsi dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

   Proyek contoh ini menyebarkan sumber daya berikut:

   - Lima Lambda fungsi
   - Amazon Simple Queue ServiceAntrian
   - Sebuah Amazon Simple Notification Service topik
   - Mesin status AWS Step Functions
   - Peran terkait AWS Identity and Access Management (IAM)

   Gambar berikut menunjukkan grafik alur kerja untuk proyek contoh fungsi Orchestrate Lambda:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

     Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi

selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

> **Important**
>
> Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

  > **Tip**
  >
  > Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

  Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

  Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

  Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

  > **Important**
  >
  > Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

Setelah semua sumber daya disediakan dan digunakan, kotak dialog Mulai eksekusi ditampilkan.

1. Pada halaman mesin Negara, pilih proyek sampel Anda.

2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

    1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

        > **Note**
        >
        > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

    2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

        Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

        > **Note**
        >
        > Jika proyek demo yang Anda gunakan berisi data input eksekusi yang telah diisi sebelumnya, gunakan input tersebut untuk menjalankan mesin status.

    3. Pilih Mulai Eksekusi.
    4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

        Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Tentang mesin negara dan pelaksanaannya

Mesin status dalam proyek sampel ini terintegrasi AWS Lambda dengan meneruskan parameter langsung ke sumber daya tersebut, menggunakan antrian Amazon SQS untuk mengelola permintaan persetujuan manusia, dan menggunakan topik Amazon SNS untuk mengembalikan hasil kueri.

Sebuah eksekusi Step Functions menerima teks JSON sebagai input dan meneruskan input yang ke status pertama dalam alur kerja. Status individu menerima data JSON sebagai input dan biasanya meneruskan data JSON sebagai output ke status berikutnya. Dalam proyek sampel ini, output dari setiap langkah diteruskan sebagai input ke langkah berikutnya dalam alur kerja. Misalnya, langkah rekomendasi Hasilkan Beli/Jual menerima output dari langkah Periksa Harga Saham sebagai masukan. Selanjutnya, output dari langkah rekomendasi Generate Beli/Sell diteruskan sebagai masukan ke langkah berikutnya, Request Human Approval, yang meniru langkah persetujuan manusia.

> **Note**
>
> Untuk melihat output yang dikembalikan secara bertahap dan input diteruskan ke langkah, buka halaman Rincian Eksekusi untuk eksekusi alur kerja Anda. Di bagian Detail langkah, lihat input dan output untuk setiap langkah yang Anda pilih dalam mode Lihat.

Untuk menerapkan langkah persetujuan manusia, Anda biasanya menjeda eksekusi alur kerja hingga token tugas dikembalikan. Dalam proyek contoh ini, pesan diteruskan ke antrean Amazon SQS, yang digunakan sebagai pemicu fungsi Lambda yang ditentukan untuk menangani fungsionalitas panggilan balik. Pesan berisi token tugas dan output yang dikembalikan oleh langkah sebelumnya. Fungsi Lambda dipanggil dengan muatan pesan. Eksekusi alur kerja dijeda hingga menerima token tugas kembali dengan panggilan API. `SendTaskSuccess` Untuk informasi selengkapnya tentang token tugas, lihatTunggu Panggilan Balik dengan Token Tugas.

Kode berikut untuk `StepFunctionsSample-HelloLambda-ApproveSqsLambda` fungsi ini menunjukkan cara didefinisikan untuk secara otomatis menyetujui tugas apa pun yang dikirimkan oleh antrian Amazon SQS melalui mesin status Step Functions.

Contoh kode fungsi Lambda untuk menangani fungsionalitas panggilan balik dan mengembalikan token tugas

```
exports.lambdaHandler = (event, context, callback) => {
  const stepfunctions = new aws.StepFunctions();
```

```
  // For every record in sqs queue
  for (const record of event.Records) {
      const messageBody = JSON.parse(record.body);
      const taskToken = messageBody.TaskToken;

      const params = {
          output: "\"approved\"",
          taskToken: taskToken
      };

      console.log(`Calling Step Functions to complete callback task with params
 ${JSON.stringify(params)}`);

      // Approve
      stepfunctions.sendTaskSuccess(params, (err, data) => {
          if (err) {
              console.error(err.message);
              callback(err.message);
              return;
          }
          console.log(data);
          callback(null);
      });
  }
};
```

Jelajahi melalui contoh mesin status ini untuk melihat cara Step Functions mengendalikan Lambda dan Amazon SQS.

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

```
{
    "StartAt": "Check Stock Price",
    "States": {
        "Check Stock Price": {
            "Type": "Task",
            "Resource": "arn:aws:lambda:us-
west-1:111122223333:function:StepFunctionsSample-HelloLam-
CheckStockPriceLambda-444455556666",
            "Next": "Generate Buy/Sell recommendation"
        },
```

```
        "Generate Buy/Sell recommendation": {
            "Type": "Task",
            "Resource": "arn:aws:lambda:us-
west-1:111122223333:function:StepFunctionsSample-Hello-
GenerateBuySellRecommend-123456789012",
            "ResultPath": "$.recommended_type",
            "Next": "Request Human Approval"
        },
        "Request Human Approval": {
            "Type": "Task",
            "Resource": "arn:aws:states:::sqs:sendMessage.waitForTaskToken",
            "Parameters": {
                "QueueUrl": "https://sqs.us-west-1.amazonaws.com/111122223333/
StepFunctionsSample-HelloLambda4444-5555-6666-RequestHumanApprovalSqs-777788889999",
                "MessageBody": {
                    "Input.$": "$",
                    "TaskToken.$": "$$.Task.Token"
                }
            },
            "ResultPath": null,
            "Next": "Buy or Sell?"
        },
        "Buy or Sell?": {
            "Type": "Choice",
            "Choices": [
                {
                    "Variable": "$.recommended_type",
                    "StringEquals": "buy",
                    "Next": "Buy Stock"
                },
                {
                    "Variable": "$.recommended_type",
                    "StringEquals": "sell",
                    "Next": "Sell Stock"
                }
            ]
        },
        "Buy Stock": {
            "Type": "Task",
            "Resource": "arn:aws:lambda:us-
west-1:111122223333:function:StepFunctionsSample-HelloLambda-
BuyStockLambda-000000000000",
            "Next": "Report Result"
        },
```

```
        "Sell Stock": {
            "Type": "Task",
            "Resource": "arn:aws:lambda:us-
west-1:111122223333:function:StepFunctionsSample-HelloLambda-
SellStockLambda-111111111111",
            "Next": "Report Result"
        },
        "Report Result": {
            "Type": "Task",
            "Resource": "arn:aws:states:::sns:publish",
            "Parameters": {
                "TopicArn": "arn:aws:sns:us-west-1:111122223333:StepFunctionsSample-
HelloLambda1111-2222-3333-ReportResultSnsTopic-222222222222",
                "Message": {
                    "Input.$": "$"
                }
            },
            "End": true
        }
    }
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Contoh IAM

Contoh kebijakan AWS Identity and Access Management (IAM) yang dihasilkan oleh proyek sampel ini mencakup hak istimewa paling sedikit yang diperlukan untuk mengeksekusi mesin negara dan sumber daya terkait. Kami merekomendasikan agar Anda hanya menyertakan izin yang diperlukan dalam kebijakan IAM Anda.

```
{
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:us-
west-1:111122223333:function:StepFunctionsSample-HelloLam-
CheckStockPriceLambda-444455556666",
            "Effect": "Allow"
        }
```

```
    ]
}
```

```
{
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:us-
west-1:111122223333:function:StepFunctionsSample-Hello-
GenerateBuySellRecommend-123456789012",
            "Effect": "Allow"
        }
    ]
}
```

```
{
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:us-
west-1:111122223333:function:StepFunctionsSample-HelloLambda-
BuyStockLambda-777788889999",
            "Effect": "Allow"
        }
    ]
}
```

```
{
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:us-
west-1:111122223333:function:StepFunctionsSample-HelloLambda-
SellStockLambda-000000000000",
            "Effect": "Allow"
        }
```

```
    ]
}
```

```
{
    "Statement": [
        {
            "Action": [
                "sqs:SendMessage*"
            ],
            "Resource": "arn:aws:sqs:us-west-1:111122223333:StepFunctionsSample-
HelloLambda4444-5555-6666-RequestHumanApprovalSqs-111111111111",
            "Effect": "Allow"
        }
    ]
}
```

```
{
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": "arn:aws:sns:us-west-1:111122223333:StepFunctionsSample-
HelloLambda1111-2222-3333-ReportResultSnsTopic-222222222222",
            "Effect": "Allow"
        }
    ]
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Mulai kueri Athena

Proyek sampel ini, yang didasarkan pada alur kerja standar, menunjukkan cara menggunakan Step Functions dan Amazon Athena untuk memulai kueri Athena dan mengirim pemberitahuan dengan hasil kueri.

Dalam proyek ini, Step Functions menggunakan fungsi Lambda dan AWS Glue crawler untuk menghasilkan sekumpulan data contoh. Kemudian melakukan kueri menggunakan Integrasi layanan Athena dan mengembalikan hasilnya menggunakan topik SNS.

Untuk informasi selengkapnya tentang integrasi layanan Athena dan Step Functions, lihat hal berikut:

- Menggunakan AWS Step Functions dengan layanan lain
- Panggil Athena dengan Step Functions

> **Note**
>
> Proyek sampel ini mungkin dikenakan biaya.
> Untuk AWS pengguna baru, tingkat penggunaan gratis tersedia. Pada tingkat ini, layanan akan gratis di bawah tingkat penggunaan tertentu. Untuk informasi lebih lanjut tentang AWS biaya dan Tingkat Gratis, lihat Harga Athena.

Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`Start an Athena query`** di kotak pencarian, lalu pilih Mulai Athena kueri dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

   Proyek contoh ini menyebarkan sumber daya berikut:

   - Sebuah Amazon Athena kueri
   - Sebuah Perayap AWS Glue
   - Sebuah Amazon SNS topik
   - Mesin status AWS Step Functions
   - Peran terkait AWS Identity and Access Management (IAM)

   Gambar berikut menunjukkan grafik alur kerja untuk Memulai proyek sampel Athena kueri:

5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

     Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

> **Important**
>
> Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

> **Tip**
>
> Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

> **Important**
>
> Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

1. Pada halaman mesin Negara, pilih proyek sampel Anda.
2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

   > **Note**
   >
   > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

   Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

   > **Note**
   >
   > Jika proyek demo yang Anda gunakan berisi data input eksekusi yang telah diisi sebelumnya, gunakan input tersebut untuk menjalankan mesin status.

3. Pilih Mulai Eksekusi.
4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

   Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Contoh Kode Mesin Status

Mesin status dalam proyek sampel ini terintegrasi dengan Athena AWS Lambda dan dengan meneruskan parameter langsung ke sumber daya tersebut, dan menggunakan topik SNS untuk mengembalikan hasil kueri.

Jelajahi melalui contoh mesin status ini untuk melihat cara Step Functions mengontrol Lambda dan Athena.

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

```
{
  "StartAt": "Generate example log",
  "States": {
    "Generate example log": {
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:StepFunctionsSample-
Athena-LambdaForDataGeneration-AKIAIOSFODNN7EXAMPLE",
      "Type": "Task",
      "Next": "Run Glue crawler"
    },
    "Run Glue crawler": {
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:StepFunctionsSample-
Athen-LambdaForInvokingCrawler-AKIAI44QH8DHBEXAMPLE",
      "Type": "Task",
      "Next": "Start an Athena query"
    },
    "Start an Athena query": {
      "Resource": "arn:aws:states:::athena:startQueryExecution.sync",
      "Parameters": {
        "QueryString": "SELECT * FROM \"athena-sample-project-db-wJalrXUtnFEMI\".\"log
\" limit 1",
        "WorkGroup": "stepfunctions-athena-sample-project-workgroup-wJalrXUtnFEMI"
      },
      "Type": "Task",
      "Next": "Get query results"
    },
    "Get query results": {
      "Resource": "arn:aws:states:::athena:getQueryResults",
      "Parameters": {
        "QueryExecutionId.$": "$.QueryExecution.QueryExecutionId"
      },
      "Type": "Task",
      "Next": "Send query results"
    },
    "Send query results": {
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
```

```
        "TopicArn": "arn:aws:sns:us-east-1:111122223333:StepFunctionsSample-
AthenaDataQueryd1111-2222-3333-777788889999-SNSTopic-ANPAJ2UCCR6DPCEXAMPLE",
        "Message": {
          "Input.$": "$.ResultSet.Rows"
        }
      },
      "Type": "Task",
      "End": true
    }
  }
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Contoh IAM

Contoh kebijakan AWS Identity and Access Management (IAM) yang dihasilkan oleh proyek sampel ini mencakup hak istimewa paling sedikit yang diperlukan untuk mengeksekusi mesin negara dan sumber daya terkait. Kami merekomendasikan agar Anda hanya menyertakan izin yang diperlukan dalam kebijakan IAM Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:111122223333:function:StepFunctionsSample-
Athena-LambdaForDataGeneration-AKIAIOSFODNN7EXAMPLE",
                "arn:aws:lambda:us-east-1:111122223333:function:StepFunctionsSample-
Athen-LambdaForInvokingCrawler-AKIAI44QH8DHBEXAMPLE"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
```

```
                "arn:aws:sns:us-east-1:111122223333:StepFunctionsSample-
AthenaDataQueryd1111-2222-3333-777788889999-SNSTopic-ANPAJ2UCCR6DPCEXAMPLE"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "athena:getQueryResults",
                "athena:startQueryExecution",
                "athena:stopQueryExecution",
                "athena:getQueryExecution",
                "athena:getDataCatalog"
            ],
            "Resource": [
                "arn:aws:athena:us-east-1:111122223333:workgroup/stepfunctions-athena-
sample-project-workgroup-wJalrXUtnFEMI",
                "arn:aws:athena:us-east-1:111122223333:datacatalog/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload",
                "s3:CreateBucket",
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "glue:CreateDatabase",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:UpdateDatabase",
                "glue:DeleteDatabase",
                "glue:CreateTable",
                "glue:UpdateTable",
                "glue:GetTable",
```

```
                "glue:GetTables",
                "glue:DeleteTable",
                "glue:BatchDeleteTable",
                "glue:BatchCreatePartition",
                "glue:CreatePartition",
                "glue:UpdatePartition",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition",
                "glue:DeletePartition",
                "glue:BatchDeletePartition"
            ],
            "Resource": [
                "arn:aws:glue:us-east-1:111122223333:database/*",
                "arn:aws:glue:us-east-1:111122223333:table/*",
                "arn:aws:glue:us-east-1:111122223333:catalog"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Jalankan beberapa kueri (Amazon Athena, Amazon SNS)

Proyek sampel ini menunjukkan cara menjalankan kueri Athena secara berurutan dan kemudian secara paralel, menangani kesalahan dan kemudian mengirim pemberitahuan Amazon SNS berdasarkan apakah kueri berhasil atau gagal.

Dalam proyek ini, Step Functions menggunakan mesin status untuk menjalankan kueri Athena secara serempak. Setelah hasil query dikembalikan, masukkan keadaan parallel dengan dua query Athena mengeksekusi secara paralel. Kemudian menunggu tugas berhasil atau gagal, dan mengirimkan topik Amazon SNS dengan pesan tentang apakah tugas berhasil atau gagal.

Langkah 1: Buat mesin negara dan sumber daya penyediaan

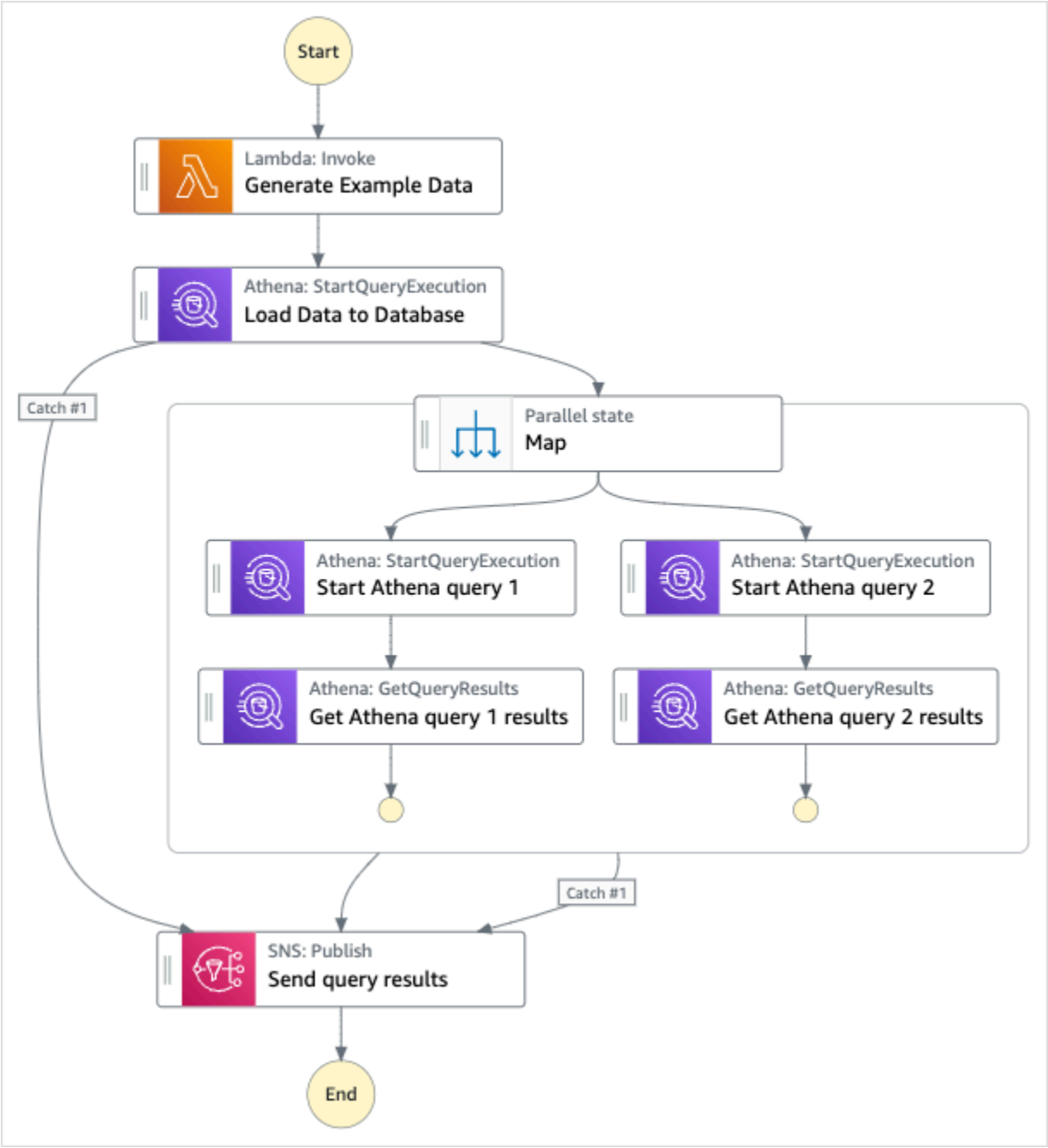
1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`Execute multiple queries`** di kotak pencarian, lalu pilih Jalankan beberapa kueri dari hasil pencarian yang dikembalikan.

3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

   Proyek contoh ini menyebarkan sumber daya berikut:

   - Amazon Athenapertanyaan
   - Sebuah Amazon SNS topik
   - Mesin status AWS Step Functions
   - Peran terkait AWS Identity and Access Management (IAM)

   Gambar berikut menunjukkan grafik alur kerja untuk proyek contoh Execute multiple query:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

     Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan

editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

> **Important**
>
> Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

  > **Tip**
  >
  > Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

  Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

  Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

  Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

  > **Important**
  >
  > Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

1. Pada halaman mesin Negara, pilih proyek sampel Anda.
2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

    1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions secara otomatis menghasilkan nama eksekusi yang unik.

        > **Note**
        >
        > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, aktivitas, dan label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

    2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

        Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

    3. Pilih Mulai Eksekusi.
    4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

        Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Contoh Kode Mesin Status

Mesin status dalam proyek sampel ini terintegrasi dengan Amazon Athena dan Amazon SNS dengan meneruskan parameter langsung ke sumber daya tersebut.

Jelajahi contoh mesin status ini untuk melihat bagaimana Step Functions mengontrol Amazon Athena dan Amazon SNS dengan menghubungkan ke Amazon Resource Name (ARN) di bidang, dan dengan `Resource Parameters` meneruskan ke API layanan.

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

```
{
    "Comment": "An example of using Athena to execute queries in sequence and parallel,
 with error handling and notifications.",
    "StartAt": "Generate Example Data",
    "States": {
      "Generate Example Data": {
        "Type": "Task",
        "Resource": "arn:aws:states:::lambda:invoke",
        "OutputPath": "$.Payload",
        "Parameters": {
          "FunctionName": "<ATHENA_FUNCTION_NAME>"
        },
        "Next": "Load Data to Database"
      },
      "Load Data to Database": {
        "Type": "Task",
        "Resource": "arn:aws:states:::athena:startQueryExecution.sync",
        "Parameters": {
          "QueryString": "<ATHENA_QUERYSTRING>",
          "WorkGroup": "<ATHENA_WORKGROUP>"
        },
        "Catch": [
          {
            "ErrorEquals": [
              "States.ALL"
            ],
            "Next": "Send query results"
          }
        ],
        "Next": "Map"
      },
      "Map": {
        "Type": "Parallel",
        "ResultSelector": {
          "Query1Result.$": "$[0].ResultSet.Rows",
          "Query2Result.$": "$[1].ResultSet.Rows"
```

```
        },
        "Catch": [
          {
            "ErrorEquals": [
              "States.ALL"
            ],
            "Next": "Send query results"
          }
        ],
        "Branches": [
          {
            "StartAt": "Start Athena query 1",
            "States": {
              "Start Athena query 1": {
                "Type": "Task",
                "Resource": "arn:aws:states:::athena:startQueryExecution.sync",
                "Parameters": {
                  "QueryString": "<ATHENA_QUERYSTRING>",
                  "WorkGroup": "<ATHENA_WORKGROUP>"
                },
                "Next": "Get Athena query 1 results"
              },
              "Get Athena query 1 results": {
                "Type": "Task",
                "Resource": "arn:aws:states:::athena:getQueryResults",
                "Parameters": {
                  "QueryExecutionId.$": "$.QueryExecution.QueryExecutionId"
                },
                "End": true
              }
            }
          },
          {
            "StartAt": "Start Athena query 2",
            "States": {
              "Start Athena query 2": {
                "Type": "Task",
                "Resource": "arn:aws:states:::athena:startQueryExecution.sync",
                "Parameters": {
                    "QueryString": "<ATHENA_QUERYSTRING>",
                    "WorkGroup": "<ATHENA_WORKGROUP>"
                },
                "Next": "Get Athena query 2 results"
              },
```

```
              "Get Athena query 2 results": {
                "Type": "Task",
                "Resource": "arn:aws:states:::athena:getQueryResults",
                "Parameters": {
                  "QueryExecutionId.$": "$.QueryExecution.QueryExecutionId"
                },
                "End": true
              }
            }
          }
        ],
        "Next": "Send query results"
      },
      "Send query results": {
        "Type": "Task",
        "Resource": "arn:aws:states:::sns:publish",
        "Parameters": {
          "Message.$": "$",
          "TopicArn": "<SNS_TOPIC_ARN>"
        },
        "End": true
      }
    }
  }
```

Contoh IAM

Contoh kebijakan AWS Identity and Access Management (IAM) yang dihasilkan oleh proyek sampel ini mencakup hak istimewa paling sedikit yang diperlukan untuk mengeksekusi mesin negara dan sumber daya terkait. Kami merekomendasikan agar Anda hanya menyertakan izin yang diperlukan dalam kebijakan IAM Anda.

AthenaStartQueryExecution

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:startQueryExecution",
                "athena:stopQueryExecution",
```

```
                "athena:getQueryExecution",
                "athena:getDataCatalog"
            ],
            "Resource": [
                "arn:aws:athena:us-east-2:123456789012:workgroup/stepfunctions-athena-
sample-project-workgroup-ztuvu9yuix",
                "arn:aws:athena:us-east-2:123456789012:datacatalog/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload",
                "s3:CreateBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:CreateDatabase",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:UpdateDatabase",
                "glue:DeleteDatabase",
                "glue:CreateTable",
                "glue:UpdateTable",
                "glue:GetTable",
                "glue:GetTables",
                "glue:DeleteTable",
                "glue:BatchDeleteTable",
                "glue:BatchCreatePartition",
                "glue:CreatePartition",
                "glue:UpdatePartition",
                "glue:GetPartition",
                "glue:GetPartitions",
```

```
                "glue:BatchGetPartition",
                "glue:DeletePartition",
                "glue:BatchDeletePartition"
            ],
            "Resource": [
                "arn:aws:glue:us-east-2:123456789012:catalog",
                "arn:aws:glue:us-east-2:123456789012:database/*",
                "arn:aws:glue:us-east-2:123456789012:table/*",
                "arn:aws:glue:us-east-2:123456789012:userDefinedFunction/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lakeformation:GetDataAccess"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

AthenaGetQueryResults

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:getQueryResults"
            ],
            "Resource": [
                "arn:aws:us-east-2:123456789012:workgroup/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
```

```
            "Resource": [
                "arn:aws:s3:::*"
            ]
        }
    ]
}
```

SNSPublish

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:us-east-2:123456789012:StepFunctionsSample-
AthenaMultipleQueriese1ec229b-5cbe-4754-a8a8-078474bac878-SNSTopic-9AID0HEJT7TH"
            ]
        }
    ]
}
```

LambdaInvokeFunction

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-2:123456789012:function:StepFunctionsSample-
Athen-LambdaForStringGeneratio-GQFQjN7mE9gl:*"
            ]
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-2:123456789012:function:StepFunctionsSample-
Athen-LambdaForStringGeneratio-GQFQjN7mE9gl"
            ]
        }
    ]
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Kueri kumpulan data besar (Amazon Athena, Amazon S3,, AWS Glue Amazon SNS)

Proyek sampel ini menunjukkan cara menelan kumpulan data besar di Amazon S3 dan mempartisinya AWS Glue melalui Crawler, lalu menjalankan kueri Amazon Athena terhadap partisi tersebut.

Dalam proyek ini, mesin status Step Functions memanggil AWS Glue crawler yang mempartisi kumpulan data besar di Amazon S3. Setelah AWS Glue crawler mengembalikan pesan sukses, alur kerja mengeksekusi kueri Athena terhadap partisi tersebut. Setelah eksekusi kueri berhasil diselesaikan, notifikasi Amazon SNS dikirim ke topik Amazon SNS.

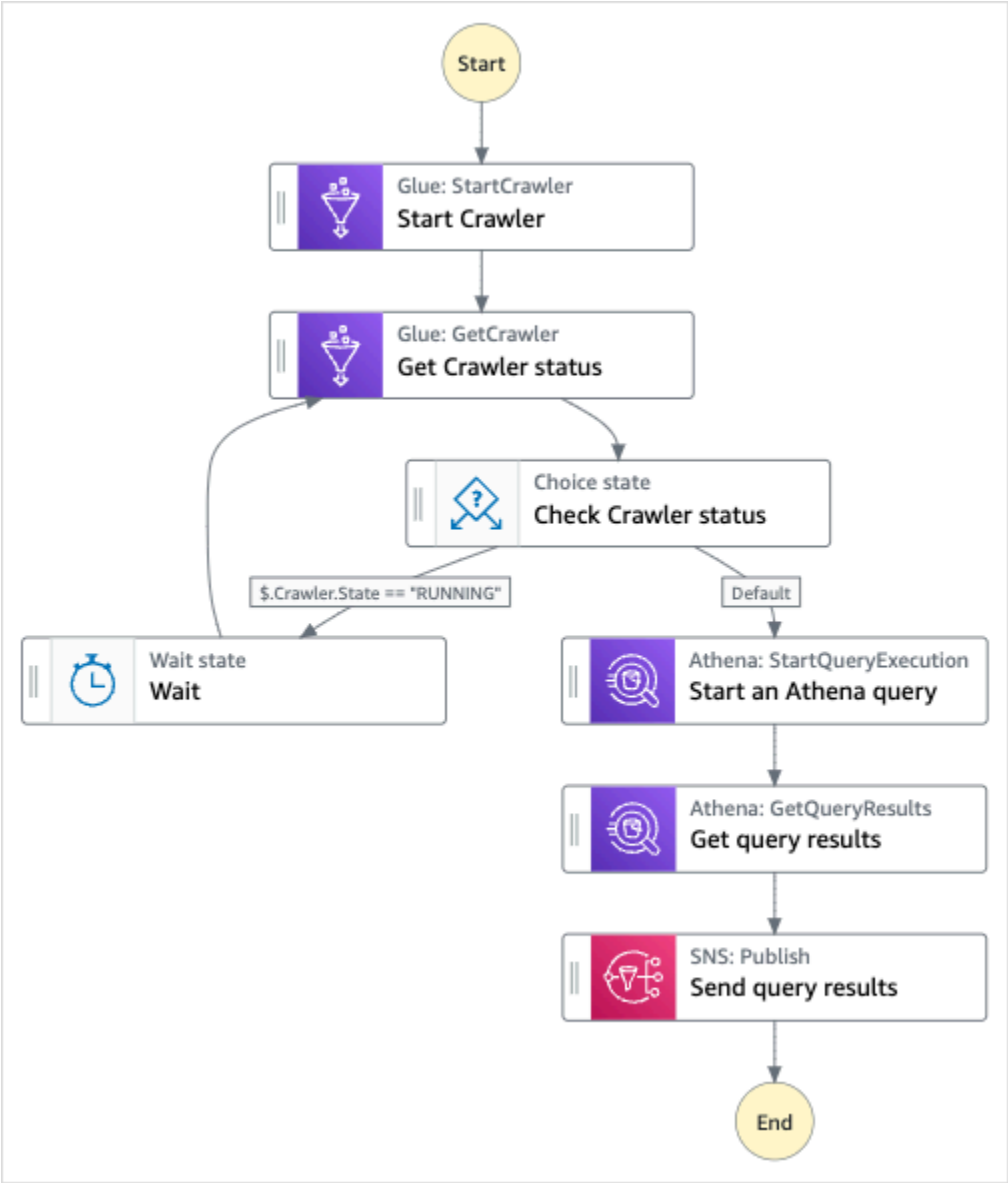
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`Query large datasets`** di kotak pencarian, lalu pilih Kueri kumpulan data besar dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

Proyek contoh ini menyebarkan sumber daya berikut:

- Amazon S3Ember
- Sebuah Perayap AWS Glue
- Sebuah Amazon SNS topik
- Mesin status AWS Step Functions
- Peran terkait AWS Identity and Access Management (IAM)

Gambar berikut menunjukkan grafik alur kerja untuk proyek sampel kumpulan data besar Query:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.

6. Lakukan salah satu hal berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

     Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

     > **Important**
     >
     > Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

   - Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

     > **Tip**
     >
     > Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

     Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

     Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

     Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

> **Important**
>
> Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

1. Pada halaman mesin Negara, pilih proyek sampel Anda.
2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

    1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions secara otomatis menghasilkan nama eksekusi yang unik.

        > **Note**
        >
        > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, aktivitas, dan label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

    2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

        Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

    3. Pilih Mulai Eksekusi.
    4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

        Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output,

dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Contoh Kode Mesin Status

Mesin status dalam proyek sampel ini terintegrasi dengan Amazon S3 AWS Glue,, Amazon Athena dan Amazon SNS dengan meneruskan parameter langsung ke sumber daya tersebut.

Jelajahi contoh mesin status ini untuk melihat bagaimana Step Functions mengontrol Amazon S3 AWS Glue, Amazon Athena, dan Amazon SNS dengan menghubungkan ke Amazon Resource Name (ARN) di lapangan, dan dengan meneruskan ke `Resource` API layanan. `Parameters`

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

```
{
    "Comment": "An example demonstrates how to ingest a large data set in Amazon S3 and
 partition it through aws Glue Crawlers, then execute Amazon Athena queries against
 that partition.",
   "StartAt": "Start Crawler",
   "States": {
     "Start Crawler": {
       "Type": "Task",
       "Next": "Get Crawler status",
       "Parameters": {
         "Name": "<GLUE_CRAWLER_NAME>"
       },
       "Resource": "arn:aws:states:::aws-sdk:glue:startCrawler"
     },
     "Get Crawler status": {
       "Type": "Task",
       "Parameters": {
         "Name": "<GLUE_CRAWLER_NAME>"
       },
       "Resource": "arn:aws:arn:aws:states:::aws-sdk:glue:getCrawler",
       "Next": "Check Crawler status"
     },
     "Check Crawler status": {
       "Type": "Choice",
       "Choices": [
         {
           "Variable": "$.Crawler.State",
```

```
            "StringEquals": "RUNNING",
            "Next": "Wait"
          }
        ],
        "Default": "Start an Athena query"
      },
      "Wait": {
        "Type": "Wait",
        "Seconds": 30,
        "Next": "Get Crawler status"
      },
      "Start an Athena query": {
        "Resource": "arn:aws:states:::athena:startQueryExecution.sync",
        "Parameters": {
            "QueryString": "<ATHENA_QUERYSTRING>",
            "WorkGroup": "<ATHENA_WORKGROUP>"
        },
        "Type": "Task",
        "Next": "Get query results"
      },
      "Get query results": {
        "Resource": "arn:aws:states:::athena:getQueryResults",
        "Parameters": {
          "QueryExecutionId.$": "$.QueryExecution.QueryExecutionId"
        },
        "Type": "Task",
        "Next": "Send query results"
      },
      "Send query results": {
        "Resource": "arn:aws:states:::sns:publish",
        "Parameters": {
          "TopicArn": "<SNS_TOPIC_ARN>",
          "Message": {
            "Input.$": "$.ResultSet.Rows"
          }
        },
        "Type": "Task",
        "End": true
      }
    }
  }
```

Contoh IAM

Contoh kebijakan AWS Identity and Access Management (IAM) yang dihasilkan oleh proyek sampel ini mencakup hak istimewa paling sedikit yang diperlukan untuk mengeksekusi mesin negara dan sumber daya terkait. Kami merekomendasikan agar Anda hanya menyertakan izin yang diperlukan dalam kebijakan IAM Anda.

AthenaGetQueryResults

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:getQueryResults"
            ],
            "Resource": [
                "arn:aws:athena:us-east-2:123456789012:workgroup/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::*"
            ]
        }
    ]
}
```

AthenaStartQueryExecution

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "athena:startQueryExecution",
                "athena:stopQueryExecution",
                "athena:getQueryExecution",
                "athena:getDataCatalog"
            ],
            "Resource": [
                "arn:aws:athena:us-east-2:123456789012:workgroup/stepfunctions-athena-
sample-project-workgroup-8v7bshiv70",
                "arn:aws:athena:us-east-2:123456789012:datacatalog/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload",
                "s3:CreateBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:CreateDatabase",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:UpdateDatabase",
                "glue:DeleteDatabase",
                "glue:CreateTable",
                "glue:UpdateTable",
                "glue:GetTable",
                "glue:GetTables",
                "glue:DeleteTable",
                "glue:BatchDeleteTable",
                "glue:BatchCreatePartition",
                "glue:CreatePartition",
                "glue:UpdatePartition",
```

```
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition",
                "glue:DeletePartition",
                "glue:BatchDeletePartition"
            ],
            "Resource": [
                "arn:aws:glue:us-east-2:123456789012:catalog",
                "arn:aws:glue:us-east-2:123456789012:database/*",
                "arn:aws:glue:us-east-2:123456789012:table/*",
                "arn:aws:glue:us-east-2:123456789012:userDefinedFunction/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lakeformation:GetDataAccess"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

SNSPublish

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:us-east-2:123456789012:StepFunctionsSample-
AthenaIngestLargeDataset92bc4949-abf8-4a1e-9236-5b7c81b3efa3-SNSTopic-8Y5ZLI5AASXV"
            ]
        }
    ]
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Tetap perbarui data (Amazon Athena, Amazon S3,) AWS Glue

Proyek contoh ini menunjukkan cara menanyakan tabel target untuk mendapatkan data terkini dengan AWS Glue Catalog, lalu memperbaruinya dengan data baru dari sumber lain menggunakan Amazon Athena.

Dalam project ini, mesin status Step Functions memanggil AWS Glue Catalog untuk memverifikasi apakah tabel target ada di Amazon S3 Bucket. Jika tidak ada tabel yang ditemukan, itu akan membuat tabel baru. Kemudian, Step Functions jalankan kueri Athena untuk menambahkan baris ke tabel target dari sumber data yang berbeda: pertama menanyakan tabel target untuk mendapatkan tanggal terbaru, lalu menanyakan tabel sumber untuk data yang lebih baru dan memasukkannya ke dalam tabel target.

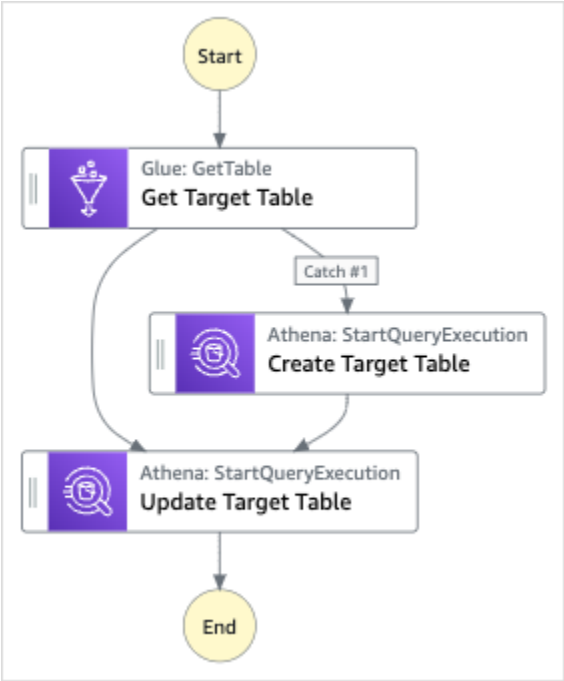
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`Keep data up to date`** di kotak pencarian, lalu pilih Perbarui data dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

   Proyek contoh ini menyebarkan sumber daya berikut:

   - Amazon S3Ember
   - Amazon Athenakueri
   - AWS Glue Data CatalogPanggilan
   - Mesin status AWS Step Functions
   - Peran terkait AWS Identity and Access Management (IAM)

Gambar berikut menunjukkan grafik alur kerja untuk Keep data up to date proyek sampel:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

     Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

     > **Important**
     >
     > Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

  > **Tip**
  >
  > Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

  Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

  Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

  Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

  > **Important**
  >
  > Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

1. Pada halaman mesin Negara, pilih proyek sampel Anda.
2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

   1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions secara otomatis menghasilkan nama eksekusi yang unik.

> **Note**
>
> Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, aktivitas, dan label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

   Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.
3. Pilih Mulai Eksekusi.
4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

   Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Contoh Kode Mesin Status

Mesin status dalam proyek sampel ini terintegrasi dengan Amazon S3 AWS Glue,, dan Amazon Athena dengan meneruskan parameter langsung ke sumber daya tersebut.

Jelajahi contoh mesin status ini untuk melihat bagaimana Step Functions mengontrol Amazon S3 AWS Glue, dan Amazon Athena dengan menghubungkan ke Amazon Resource Name (ARN) di bidang, dan dengan `Resource` `Parameters` meneruskan ke API layanan.

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

```
{
    "Comment": "An example demonstrates how to use Athena to query a target table to
 get current data, then update it with new data from other sources.",
    "StartAt": "Get Target Table",
```

```
    "States": {
      "Get Target Table": {
        "Type": "Task",
        "Parameters": {
          "DatabaseName": "<GLUE_DATABASE_NAME>",
          "Name": "target"
        },
        "Catch": [
          {
            "ErrorEquals": [
              "Glue.EntityNotFoundException"
            ],
            "Next": "Create Target Table"
          }
        ],
        "Resource": "arn:aws:states:::aws-sdk:glue:getTable",
        "Next": "Update Target Table"
      },
      "Create Target Table": {
        "Resource": "arn:aws:states:::athena:startQueryExecution.sync",
        "Parameters": {
            "QueryString": "<ATHENA_QUERYSTRING>",
            "WorkGroup": "<ATHENA_WORKGROUP>"
        },
        "Type": "Task",
        "Next": "Update Target Table"
      },
      "Update Target Table": {
        "Resource": "arn:aws:states:::athena:startQueryExecution.sync",
        "Parameters": {
            "QueryString": "<ATHENA_QUERYSTRING>",
            "WorkGroup": "<ATHENA_WORKGROUP>"
        },
        "Type": "Task",
        "End": true
      }
    }
  }
```

Contoh IAM

Contoh kebijakan AWS Identity and Access Management (IAM) yang dihasilkan oleh proyek sampel ini mencakup hak istimewa paling sedikit yang diperlukan untuk mengeksekusi mesin negara dan

sumber daya terkait. Kami merekomendasikan agar Anda hanya menyertakan izin yang diperlukan dalam kebijakan IAM Anda.

AthenaStartQueryExecution

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:startQueryExecution",
                "athena:stopQueryExecution",
                "athena:getQueryExecution",
                "athena:getDataCatalog"
            ],
            "Resource": [
                "arn:aws:athena:us-east-2:123456789012:workgroup/stepfunctions-athena-
sample-project-workgroup-26ujlyawxg",
                "arn:aws:athena:us-east-2:123456789012:datacatalog/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload",
                "s3:CreateBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:CreateDatabase",
                "glue:GetDatabase",
```

```
                "glue:GetDatabases",
                "glue:UpdateDatabase",
                "glue:DeleteDatabase",
                "glue:CreateTable",
                "glue:UpdateTable",
                "glue:GetTable",
                "glue:GetTables",
                "glue:DeleteTable",
                "glue:BatchDeleteTable",
                "glue:BatchCreatePartition",
                "glue:CreatePartition",
                "glue:UpdatePartition",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition",
                "glue:DeletePartition",
                "glue:BatchDeletePartition"
            ],
            "Resource": [
                "arn:aws::glue:us-east-2:123456789012:catalog",
                "arn:aws::glue:us-east-2:123456789012:database/*",
                "arn:aws::glue:us-east-2:123456789012:table/*",
                "arn:aws::glue:us-east-2:123456789012:userDefinedFunction/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lakeformation:GetDataAccess"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Mengelola klaster Amazon EKS

Proyek sampel ini menunjukkan cara menggunakan Step Functions dan Amazon Elastic Kubernetes Service untuk membuat klaster Amazon EKS dengan grup simpul, menjalankan tugas di Amazon EKS, lalu memeriksa hasilnya. Setelah selesai, grup simpul dan klaster Amazon EKS akan dihapus.

Untuk informasi selengkapnya tentang Step Functions dan integrasi layanan Step Functions, lihat hal berikut:

- Menggunakan AWS Step Functions dengan layanan lain
- Panggil Amazon EKS dengan Step Functions

> **Note**
>
> Proyek sampel ini mungkin dikenakan biaya.
> Untuk AWS pengguna baru, tingkat penggunaan gratis tersedia. Pada tingkat ini, layanan akan gratis di bawah tingkat penggunaan tertentu. Untuk informasi selengkapnya tentang AWS biaya dan Tingkat Gratis, lihat Harga Amazon EKS.

Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan

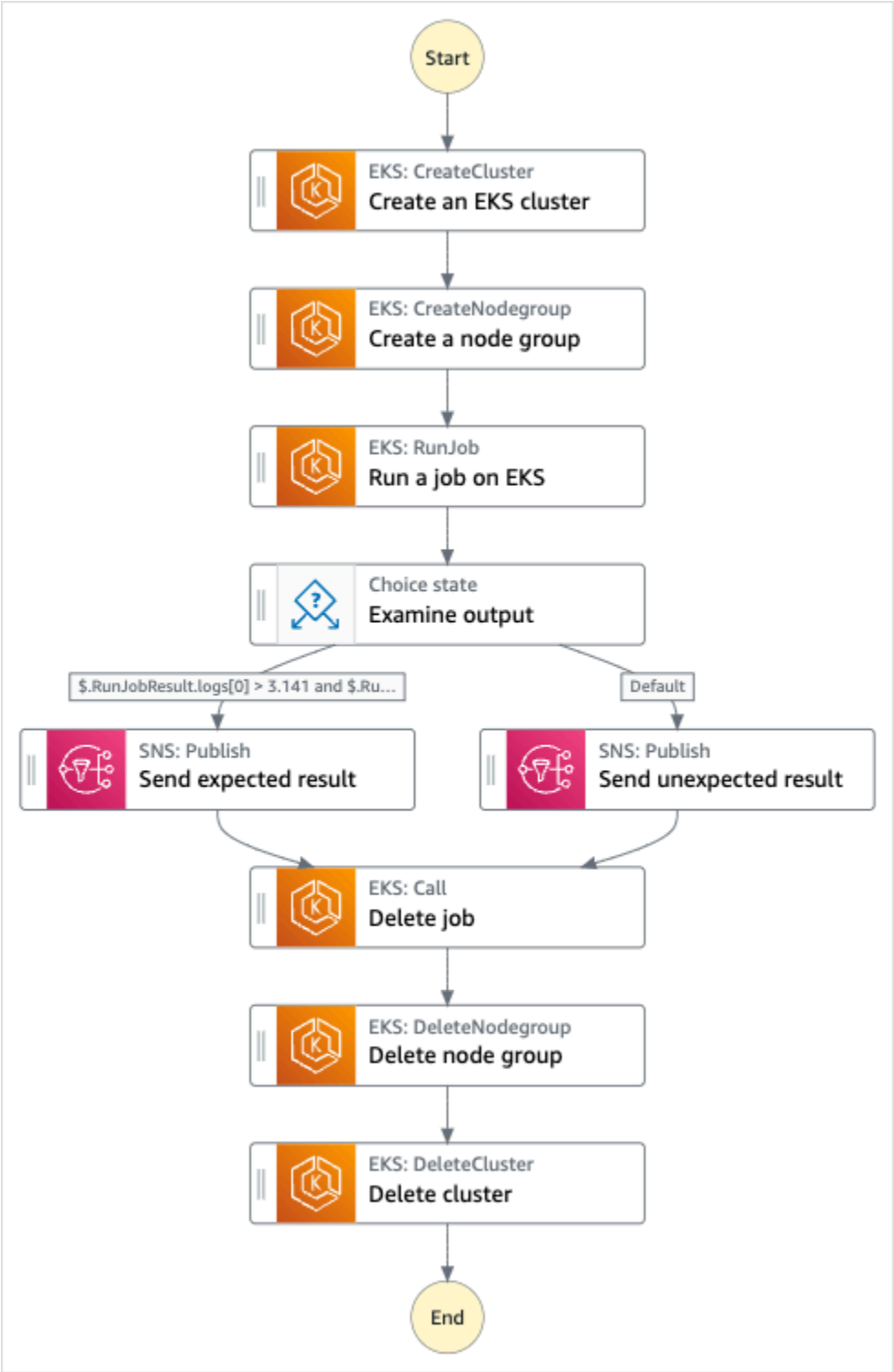
1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`Manage an EKS cluster`** di kotak pencarian, lalu pilih Kelola kluster EKS dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

   Proyek contoh ini menyebarkan sumber daya berikut:

   - Sebuah Amazon Elastic Kubernetes Service cluster
   - Sebuah Amazon SNS topik
   - Mesin status AWS Step Functions

- Peran terkait AWS Identity and Access Management (IAM)

Gambar berikut menunjukkan grafik alur kerja untuk Mengelola proyek sampel kluster EKS:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.

6. Lakukan salah satu hal berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

     Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

     > **Important**
     >
     > Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

   - Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

     > **Tip**
     >
     > Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

     Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

     Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

     Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

> **Important**
>
> Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

1. Pada halaman mesin Negara, pilih proyek sampel Anda.
2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

    1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

        > **Note**
        >
        > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

    2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

        Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

        > **Note**
        >
        > Jika proyek demo yang Anda gunakan berisi data input eksekusi yang telah diisi sebelumnya, gunakan input tersebut untuk menjalankan mesin status.

    3. Pilih Mulai Eksekusi.

4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

   Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Contoh Kode Mesin Status

Mesin status dalam proyek sampel ini terintegrasi dengan Amazon EKS dengan membuat klaster dan grup simpul Amazon EKS, dan menggunakan topik SNS untuk mengembalikan hasil.

Jelajahi contoh mesin status ini untuk melihat cara Step Functions mengelola klaster dan grup simpul Amazon EKS.

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

```
{
  "Comment": "An example of the Amazon States Language for running Amazon EKS Cluster",
  "StartAt": "Create an EKS cluster",
  "States": {
    "Create an EKS cluster": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:createCluster.sync",
      "Parameters": {
        "Name": "ExampleCluster",
        "ResourcesVpcConfig": {
          "SubnetIds": [
            "subnet-0aacf887d9f00e6a7",
            "subnet-0e5fc41e7507194ab"
          ]
        },
        "RoleArn": "arn:aws:iam::111122223333:role/StepFunctionsSample-EKSClusterManag-
EKSServiceRole-ANPAJ2UCCR6DPCEXAMPLE"
      },
      "Retry": [{
        "ErrorEquals": [ "States.ALL" ],
        "IntervalSeconds": 30,
```

```
        "MaxAttempts": 2,
        "BackoffRate": 2
      }],
      "ResultPath": "$.eks",
      "Next": "Create a node group"
    },
    "Create a node group": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:createNodegroup.sync",
      "Parameters": {
        "ClusterName": "ExampleCluster",
        "NodegroupName": "ExampleNodegroup",
        "NodeRole": "arn:aws:iam::111122223333:role/StepFunctionsSample-EKSClusterMan-
NodeInstanceRole-ANPAJ2UCCR6DPCEXAMPLE",
        "Subnets": [
          "subnet-0aacf887d9f00e6a7",
          "subnet-0e5fc41e7507194ab"]
      },
      "Retry": [{
        "ErrorEquals": [ "States.ALL" ],
        "IntervalSeconds": 30,
        "MaxAttempts": 2,
        "BackoffRate": 2
      }],
      "ResultPath": "$.nodegroup",
      "Next": "Run a job on EKS"
    },
    "Run a job on EKS": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:runJob.sync",
      "Parameters": {
        "ClusterName": "ExampleCluster",
        "CertificateAuthority.$": "$.eks.Cluster.CertificateAuthority.Data",
        "Endpoint.$": "$.eks.Cluster.Endpoint",
        "LogOptions": {
          "RetrieveLogs": true
        },
        "Job": {
          "apiVersion": "batch/v1",
          "kind": "Job",
          "metadata": {
            "name": "example-job"
          },
          "spec": {
```

```
            "backoffLimit": 0,
            "template": {
              "metadata": {
                "name": "example-job"
              },
              "spec": {
                "containers": [
                  {
                    "name": "pi-20",
                    "image": "perl",
                    "command": [
                      "perl"
                    ],
                    "args": [
                      "-Mbignum=bpi",
                      "-wle",
                      "print '{ ' . '\"pi\": '. bpi(20) . ' }';"
                    ]
                  }
                ],
                "restartPolicy": "Never"
              }
            }
          }
        }
      },
      "ResultSelector": {
        "status.$": "$.status",
        "logs.$": "$.logs..pi"
      },
      "ResultPath": "$.RunJobResult",
      "Next": "Examine output"
    },
    "Examine output": {
      "Type": "Choice",
      "Choices": [
        {
          "And": [
            {
              "Variable": "$.RunJobResult.logs[0]",
              "NumericGreaterThan": 3.141
            },
            {
              "Variable": "$.RunJobResult.logs[0]",
```

```
              "NumericLessThan": 3.142
            }
          ],
          "Next": "Send expected result"
        }
      ],
      "Default": "Send unexpected result"
    },
    "Send expected result": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
        "TopicArn": "arn:aws:sns:sa-east-1:111122223333:StepFunctionsSample-
EKSClusterManagement123456789012-SNSTopic-ANPAJ2UCCR6DPCEXAMPLE",
        "Message": {
          "Input.$": "States.Format('Saw expected value for pi: {}',
 $.RunJobResult.logs[0])"
        }
      },
      "ResultPath": "$.SNSResult",
      "Next": "Delete job"
    },
    "Send unexpected result": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
        "TopicArn": "arn:aws:sns:sa-east-1:111122223333:StepFunctionsSample-
EKSClusterManagement123456789012-SNSTopic-ANPAJ2UCCR6DPCEXAMPLE",
        "Message": {
          "Input.$": "States.Format('Saw unexpected value for pi: {}',
 $.RunJobResult.logs[0])"
        }
      },
      "ResultPath": "$.SNSResult",
      "Next": "Delete job"
    },
    "Delete job": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:call",
      "Parameters": {
        "ClusterName": "ExampleCluster",
        "CertificateAuthority.$": "$.eks.Cluster.CertificateAuthority.Data",
        "Endpoint.$": "$.eks.Cluster.Endpoint",
        "Method": "DELETE",
```

```
        "Path": "/apis/batch/v1/namespaces/default/jobs/example-job"
      },
      "ResultSelector": {
        "status.$": "$.ResponseBody.status"
      },
      "ResultPath": "$.DeleteJobResult",
      "Next": "Delete node group"
    },
    "Delete node group": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:deleteNodegroup.sync",
      "Parameters": {
        "ClusterName": "ExampleCluster",
        "NodegroupName": "ExampleNodegroup"
      },
      "Next": "Delete cluster"
    },
    "Delete cluster": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:deleteCluster.sync",
      "Parameters": {
        "Name": "ExampleCluster"
      },
      "End": true
    }
  }
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Contoh IAM

Contoh kebijakan AWS Identity and Access Management (IAM) yang dihasilkan oleh proyek sampel ini mencakup hak istimewa paling sedikit yang diperlukan untuk mengeksekusi mesin negara dan sumber daya terkait. Kami merekomendasikan agar Anda hanya menyertakan izin yang diperlukan dalam kebijakan IAM Anda.

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

```
            "Effect": "Allow",
            "Action": [
                "eks:CreateCluster"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster",
                "eks:DeleteCluster"
            ],
            "Resource": "arn:aws:eks:sa-east-1:111122223333:cluster/*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "arn:aws:iam::111122223333:role/StepFunctionsSample-EKSClusterManag-
EKSServiceRole-ANPAJ2UCCR6DPCEXAMPLE"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "eks.amazonaws.com"
                }
            }
        }
    ]
}
```

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:sa-east-1:111122223333:StepFunctionsSample-
EKSClusterManagement123456789012-SNSTopic-ANPAJ2UCCR6DPCEXAMPLE"
            ]
        }
```

```
    ]
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Buat panggilan ke API Gateway

Proyek sampel ini menunjukkan cara menggunakan Step Functions untuk membuat panggilan ke API Gateway dan memeriksa apakah panggilan berhasil.

Untuk informasi selengkapnya tentang integrasi layanan API Gateway dan Step Functions, lihat hal berikut:

- Menggunakan AWS Step Functions dengan layanan lain
- Panggil API Gateway dengan Step Functions

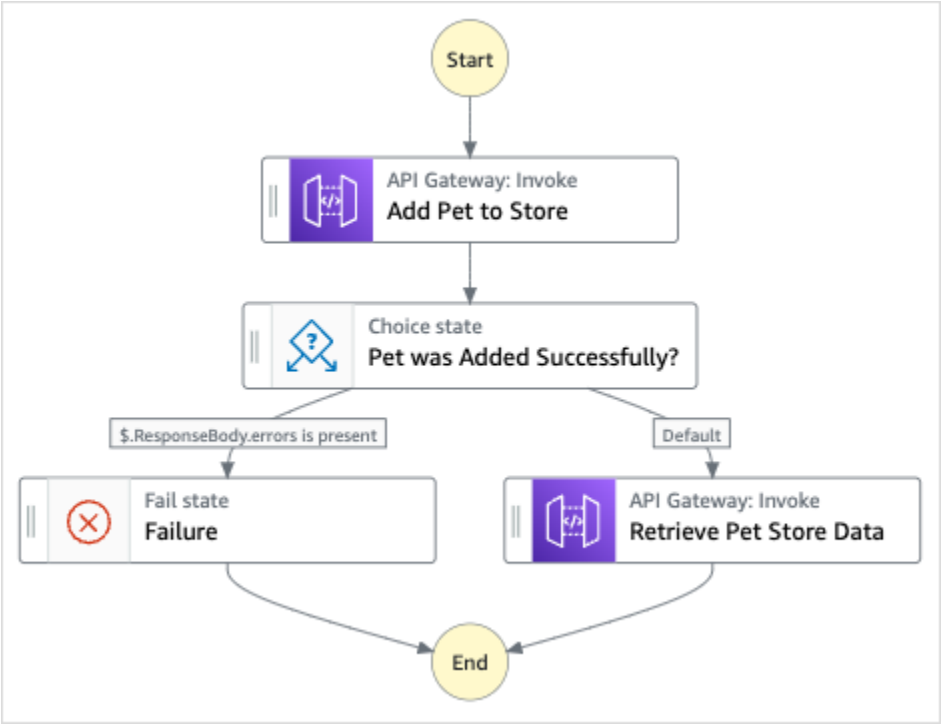
Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`Make a call to API Gateway`** di kotak pencarian, lalu pilih Buat panggilan API Gateway dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

   Proyek contoh ini menyebarkan sumber daya berikut:

   - Amazon API GatewayREST API yang dipanggil oleh state machine.
   - Mesin status AWS Step Functions
   - Peran terkait AWS Identity and Access Management (IAM)

   Gambar berikut menunjukkan grafik alur kerja untuk proyek Membuat panggilan ke API Gateway sampel:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

     Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

     > **Important**
     >
     > Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

  > **Tip**
  >
  > Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

  Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

  Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

  Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

  > **Important**
  >
  > Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

1. Pada halaman mesin Negara, pilih proyek sampel Anda.
2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

   1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

> **Note**
>
> Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

   Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

   > **Note**
   >
   > Jika proyek demo yang Anda gunakan berisi data input eksekusi yang telah diisi sebelumnya, gunakan input tersebut untuk menjalankan mesin status.

3. Pilih Mulai Eksekusi.
4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

   Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Contoh Kode Mesin Status

Mesin status dalam proyek sample ini terintegrasi dengan API Gateway dengan memanggil API REST API Gateway dan meneruskan parameter yang diperlukan.

Telusuri melalui mesin status contoh ini untuk melihat cara Step Functions berinteraksi dengan API Gateway.

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

```
{
  "Comment": "Calling APIGW REST Endpoint",
  "StartAt": "Add Pet to Store",
  "States": {
    "Add Pet to Store": {
      "Type": "Task",
      "Resource": "arn:aws:states:::apigateway:invoke",
      "Parameters": {
        "ApiEndpoint": "<POST_PETS_API_ENDPOINT>",
        "Method": "POST",
        "Stage": "default",
        "Path": "pets",
        "RequestBody.$": "$.NewPet",
        "AuthType": "IAM_ROLE"
      },
      "ResultSelector": {
        "ResponseBody.$": "$.ResponseBody"
      },
      "Next": "Pet was Added Successfully?"
    },
    "Pet was Added Successfully?": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.ResponseBody.errors",
          "IsPresent": true,
          "Next": "Failure"
        }
      ],
      "Default": "Retrieve Pet Store Data"
    },
    "Failure": {
      "Type": "Fail"
    },
    "Retrieve Pet Store Data": {
      "Type": "Task",
      "Resource": "arn:aws:states:::apigateway:invoke",
      "Parameters": {
        "ApiEndpoint": "<GET_PETS_API_ENDPOINT>",
        "Method": "GET",
        "Stage": "default",
        "Path": "pets",
        "AuthType": "IAM_ROLE"
```

```
      },
      "ResultSelector": {
        "Pets.$": "$.ResponseBody"
      },
      "ResultPath": "$.ExistingPets",
      "End": true
    }
  }
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Contoh IAM

Contoh kebijakan AWS Identity and Access Management (IAM) yang dihasilkan oleh proyek sampel ini mencakup hak istimewa paling sedikit yang diperlukan untuk mengeksekusi mesin negara dan sumber daya terkait. Kami merekomendasikan agar Anda hanya menyertakan izin yang diperlukan dalam kebijakan IAM Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "execute-api:Invoke"
            ],
            "Resource": [
                "arn:aws:execute-api:us-west-1:111122223333:c8hqe4kdg5/default/GET/
pets",
                "arn:aws:execute-api:us-west-1:111122223333:c8hqe4kdg5/default/POST/
pets"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Panggil layanan mikro yang berjalan di Fargate menggunakan integrasi API Gateway

Proyek contoh ini menunjukkan cara menggunakan Step Functions untuk melakukan panggilan ke API Gateway agar dapat berinteraksi dengan layanan AWS Fargate, dan juga untuk memeriksa apakah panggilan berhasil.

Untuk informasi selengkapnya tentang integrasi layanan API Gateway dan Step Functions, lihat hal berikut:

- Menggunakan AWS Step Functions dengan layanan lain
- Panggil API Gateway dengan Step Functions

> **Note**
>
> Proyek sampel ini mungkin dikenakan biaya.
> Untuk AWS pengguna baru, tingkat penggunaan gratis tersedia. Pada tingkat ini, layanan akan gratis di bawah tingkat penggunaan tertentu. Untuk informasi selengkapnya tentang AWS biaya dan Tingkat Gratis, lihat Harga.

Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan

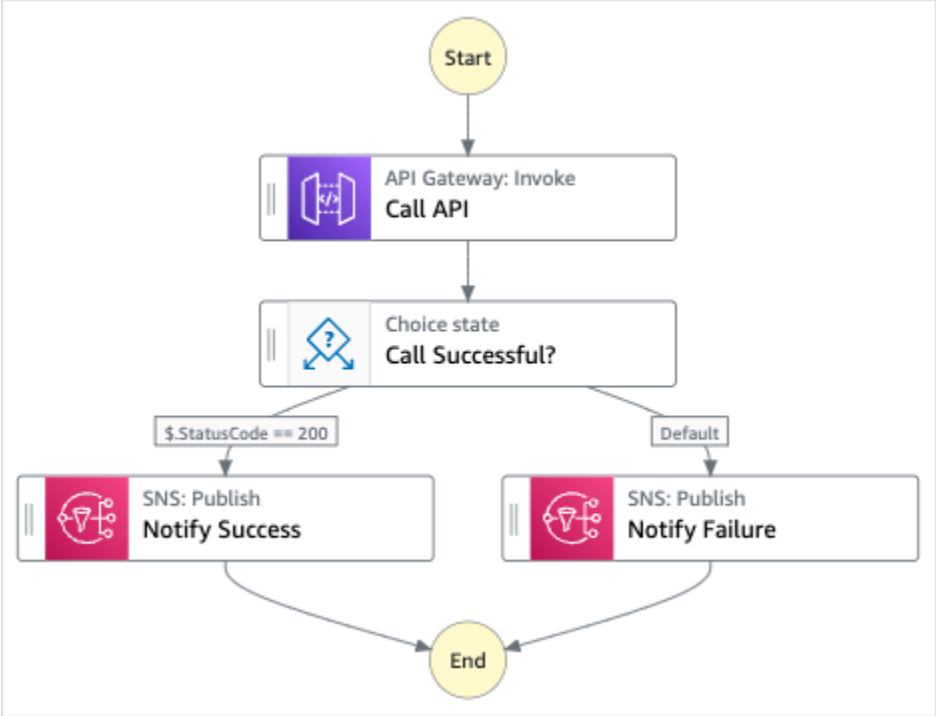
1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`Call a microservice with API Gateway`** di kotak pencarian, lalu pilih Panggil layanan mikro dengan API Gateway dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

   Proyek contoh ini menyebarkan sumber daya berikut:

   - Amazon API GatewayHTTP API yang dipanggil oleh state machine.
   - Sebuah Amazon API Gateway Amazon VPC Tautan.

- Sesi Amazon Virtual Private Cloud.
- Sesi Application Load Balancer.
- Sebuah Fargate cluster.
- Sebuah Amazon SNS topik
- Mesin AWS Step Functions negara
- Peran terkait AWS Identity and Access Management (IAM)
- Beberapa layanan tambahan yang diperlukan untuk memungkinkan sumber daya ini untuk bekerja sama.

Gambar berikut menunjukkan grafik alur kerja untuk Panggil layanan mikro dengan proyek API Gateway sampel:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

     Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan

editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

> **Important**
>
> Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

  > **Tip**
  >
  > Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

  Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

  Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

  Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

  > **Important**
  >
  > Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

1. Pada halaman mesin Negara, pilih proyek sampel Anda.
2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

    1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

        > **Note**
        >
        > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

    2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

        Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

        > **Note**
        >
        > Jika proyek demo yang Anda gunakan berisi data input eksekusi yang telah diisi sebelumnya, gunakan input tersebut untuk menjalankan mesin status.

    3. Pilih Mulai Eksekusi.
    4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

        Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Contoh Kode Mesin Status

Mesin status dalam proyek sampel ini terintegrasi dengan API Gateway dengan memanggil API HTTP API Gateway yang terhubung ke layanan di Fargate. Ini di-host di subnet privat, dan diakses melalui application load balancer privat.

Telusuri melalui mesin status contoh ini untuk melihat cara Step Functions berinteraksi dengan API Gateway dan mengembalikan hasil.

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

```
{
  "Comment": "Calling APIGW HTTP Endpoint",
  "StartAt": "Call API",
  "States": {
    "Call API": {
      "Type": "Task",
      "Resource": "arn:<PARTITION>:states:::apigateway:invoke",
      "Parameters": {
        "ApiEndpoint": "<API_ENDPOINT>",
        "Method": "GET",
        "AuthType": "IAM_ROLE"
      },
      "Next": "Call Successful?"
    },
    "Call Successful?": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.StatusCode",
          "NumericEquals": 200,
          "Next": "Notify Success"
        }
      ],
      "Default": "Notify Failure"
    },
    "Notify Success": {
      "Type": "Task",
      "Resource": "arn:<PARTITION>:states:::sns:publish",
      "Parameters": {
        "Message": "Call was successful",
        "TopicArn": "<SNS_TOPIC_ARN>"
```

```
      },
      "End": true
    },
    "Notify Failure": {
      "Type": "Task",
      "Resource": "arn:<PARTITION>:states:::sns:publish",
      "Parameters": {
        "Message": "Call was not successful",
        "TopicArn": "<SNS_TOPIC_ARN>"
      },
      "End": true
    }
  }
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Contoh IAM

Contoh kebijakan AWS Identity and Access Management (IAM) yang dihasilkan oleh proyek sampel ini mencakup hak istimewa paling sedikit yang diperlukan untuk mengeksekusi mesin negara dan sumber daya terkait. Kami merekomendasikan agar Anda hanya menyertakan izin yang diperlukan dalam kebijakan IAM Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:us-east-1:111122223333:apigw-ecs-sample-2000-
SNSTopic-444455556666"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "execute-api:Invoke"
            ],
```

```
            "Resource": [
                "arn:aws:execute-api:us-east-1:111122223333:444444444444/*/GET/*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

```
{
    "Statement": [
        {
            "Action": [
                "ec2:AttachNetworkInterface",
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:Describe*",
                "ec2:DetachNetworkInterface",
                "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
                "elasticloadbalancing:DeregisterTargets",
                "elasticloadbalancing:Describe*",
                "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
                "elasticloadbalancing:RegisterTargets"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

```
{
    "Statement": [
        {
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*",
```

```
            "Effect": "Allow"
        }
    ]
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Kirim acara khusus ke EventBridge

Proyek contoh ini menunjukkan cara menggunakan Step Functions untuk mengirim peristiwa khusus ke bus acara yang cocok dengan aturan dengan beberapa target (Amazon EventBridge,, Amazon Simple Notification Service AWS Lambda, Amazon Simple Queue Service).

Untuk informasi selengkapnya tentang Step Functions dan integrasi layanan Step Functions, lihat hal berikut:

- Menggunakan AWS Step Functions dengan layanan lain
- Panggilan EventBridge dengan Step Functions

> **Note**
>
> Proyek sampel ini mungkin dikenakan biaya.
> Untuk AWS pengguna baru, tingkat penggunaan gratis tersedia. Pada tingkat ini, layanan akan gratis di bawah tingkat penggunaan tertentu. Untuk informasi selengkapnya tentang AWS biaya dan Tingkat Gratis, lihat EventBridge Harga.

Langkah 1: Buat mesin negara dan sumber daya penyediaan

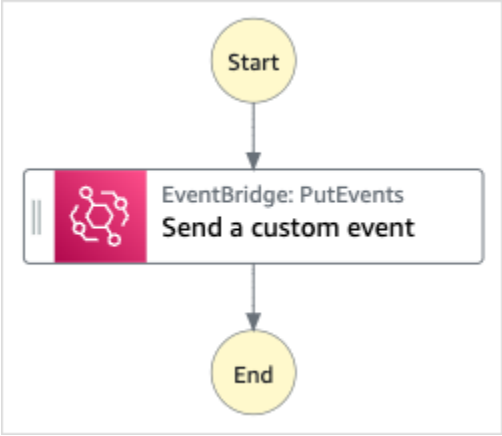
1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`Send a custom event to EventBridge`** di kotak pencarian, lalu pilih Kirim acara khusus EventBridge dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke

Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

Proyek contoh ini menyebarkan sumber daya berikut:

- Sebuah Amazon EventBridge acara
- Sebuah Amazon SNS topik
- Amazon SQSAntrian
- Sebuah Lambda fungsi
- Mesin AWS Step Functions negara
- Peran terkait AWS Identity and Access Management (IAM)

Gambar berikut menunjukkan grafik alur kerja untuk Kirim acara kustom ke proyek EventBridge sampel:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

     Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

> **Important**
>
> Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

> **Tip**
>
> Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

> **Important**
>
> Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

1. Pada halaman mesin Negara, pilih proyek sampel Anda.
2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

   > **Note**
   >
   > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

   Jika Anda memilih untuk menjalankan demo, Anda tidak perlu memberikan input eksekusi apapun.

   > **Note**
   >
   > Jika proyek demo yang Anda gunakan berisi data input eksekusi yang telah diisi sebelumnya, gunakan input tersebut untuk menjalankan mesin status.

3. Pilih Mulai Eksekusi.
4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

   Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Contoh Kode Mesin Status

Mesin status dalam proyek sampel ini terintegrasi EventBridge dengan mengirimkan acara khusus ke bus EventBridge acara. Acara yang dikirim ke bus acara cocok dengan EventBridge aturan yang memicu fungsi Lambda yang mengirim pesan ke topik Amazon SNS dan antrean Amazon SQS.

Jelajahi contoh mesin status ini untuk melihat bagaimana Step Functions mengelola EventBridge.

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

```
{
  "Comment": "An example of the Amazon States Language for sending a custom event to
 Amazon EventBridge",
  "StartAt": "Send a custom event",
  "States": {
    "Send a custom event": {
      "Resource": "arn:<PARTITION>:states:::events:putEvents",
      "Type": "Task",
      "Parameters": {
        "Entries": [{
          "Detail": {
            "Message": "Hello from Step Functions!"
          },
          "DetailType": "MessageFromStepFunctions",
          "EventBusName": "<EVENT_BUS_NAME>",
          "Source": "my.statemachine"
        }]
      },
      "End": true
    }
  }
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Contoh IAM

Contoh kebijakan AWS Identity and Access Management (IAM) yang dihasilkan oleh proyek sampel ini mencakup hak istimewa paling sedikit yang diperlukan untuk mengeksekusi mesin negara dan sumber daya terkait. Kami merekomendasikan agar Anda hanya menyertakan izin yang diperlukan dalam kebijakan IAM Anda.

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

```
            "Action": [
                "events:PutEvents"
            ],
            "Resource": [
                "arn:aws:events:us-east-1:1234567890:event-bus/stepfunctions-
sampleproject-eventbus"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Memanggil Alur Kerja Express Sinkron

Sampel proyek ini menunjukkan cara menjalankan Alur Kerja Express Sinkron melalui Amazon API Gateway untuk mengelola basis data karyawan.

Dalam proyek ini, Step Functions menggunakan titik akhir API Gateway untuk memulai Step Functions Alur Kerja Express Sinkron. Step Functions Alur Kerja Express Sinkron kemudian menggunakan DynamoDB untuk mencari, menambah, dan menghapus karyawan dalam basis data karyawan.

Untuk informasi selengkapnya tentang Step Functions Alur Kerja Express Sinkron, lihat Alur kerja Ekspres Sinkron dan Tidak Sinkron.

> **Note**
>
> Proyek sampel ini mungkin dikenakan biaya.
> Untuk AWS pengguna baru, tingkat penggunaan gratis tersedia. Pada tingkat ini, layanan akan gratis di bawah tingkat penggunaan tertentu. Untuk informasi selengkapnya tentang AWS biaya dan Tingkat Gratis, lihat Harga Step Functions.

Langkah 1: Buat mesin negara dan sumber daya penyediaan

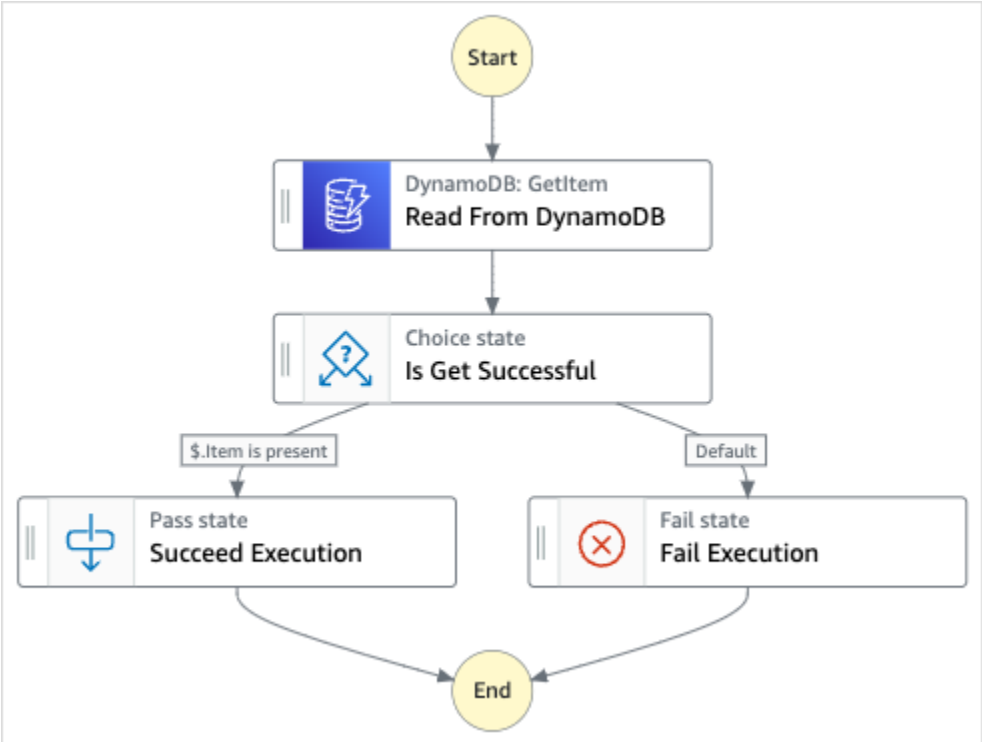
1. Buka Konsol Step Functions dan pilih Buat mesin status.

2. Ketik **`Invoke Synchronous Express Workflows through API Gateway`** di kotak pencarian, lalu pilih Invoke Synchronous Express Workflows melalui API Gateway dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

   Proyek contoh ini menyebarkan sumber daya berikut:

   - Amazon API GatewayHTTPS API yang disebut oleh mesin negara.
   - Sebuah Amazon DynamoDB meja.
   - Tiga mesin AWS Step Functions negara.
   - Peran terkait AWS Identity and Access Management (IAM).

   Gambar berikut menunjukkan grafik alur kerja untuk Invoke Synchronous Express Workflows melalui proyek sampel: API Gateway



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.

6. Lakukan salah satu hal berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

     Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

     > **Important**
     >
     > Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

   - Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

     > **Tip**
     >
     > Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

     Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

     Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

     Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

> **Important**
>
> Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

1. Pada halaman mesin Negara, pilih proyek sampel Anda.
2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

    1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

        > **Note**
        >
        > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

    2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

        Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

        > **Note**
        >
        > Jika proyek demo yang Anda gunakan berisi data input eksekusi yang telah diisi sebelumnya, gunakan input tersebut untuk menjalankan mesin status.

    3. Pilih Mulai Eksekusi.

4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

   Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Contoh Kode Mesin Status

Mesin status dalam proyek sampel ini terintegrasi dengan API Gateway dan DynamoDB menggunakan API Gateway untuk menjalankan Alur Kerja Sinkron Express, yang kemudian memperbarui atau membaca dari basis data karyawan menggunakan DynamoDB.

Jelajahi mesin status contoh ini untuk melihat bagaimana Step Functions membaca dari DynamoDB untuk mengambil informasi karyawan.

Untuk memahami selengkapnya tentang bagaimana untuk menginvoke Step Functions menggunakan API Gateway, lihat berikut.

- Panggil API Gateway dengan Step Functions
- Cara menginvoke Gateway privat dalam Panduan Developer API Gateway.

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

```
{
  "Comment": "This state machine returns an employee entry from DynamoDB",
  "StartAt": "Read From DynamoDB",
  "States": {
    "Read From DynamoDB": {
      "Type": "Task",
      "Resource": "arn:aws:states:::dynamodb:getItem",
      "Parameters": {
        "TableName": "StepFunctionsSample-
SynchronousExpressWorkflowAKIAIOSFODNN7EXAMPLE-DynamoDBTable-ANPAJ2UCCR6DPCEXAMPLE",
        "Key": {
          "EmployeeId": {"S.$": "$.employee"}
```

```
        }
      },
      "Retry": [
        {
          "ErrorEquals": [
            "DynamoDB.AmazonDynamoDBException"
          ],
          "IntervalSeconds": 3,
          "MaxAttempts": 2,
          "BackoffRate": 1.5
        }
      ],
      "Next": "Is Get Successful"
    },
    "Is Get Successful": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.Item",
          "IsPresent": true,
          "Next": "Succeed Execution"
        }
      ],
      "Default": "Fail Execution"
    },
    "Succeed Execution": {
      "Type": "Pass",
      "Parameters" : {
        "employee.$": "$.Item.EmployeeId.S",
        "jobTitle.$": "$.Item.JobTitle.S"
      },
      "End": true
    },
    "Fail Execution": {
      "Type": "Fail",
      "Error": "EmployeeDoesNotExist"
    }
  }
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Contoh IAM

Contoh kebijakan AWS Identity and Access Management (IAM) yang dihasilkan oleh proyek sampel ini mencakup hak istimewa paling sedikit yang diperlukan untuk mengeksekusi mesin negara dan sumber daya terkait. Kami merekomendasikan agar Anda hanya menyertakan izin yang diperlukan dalam kebijakan IAM Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries",
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups"
            ],
            "Resource": "*"
        }
    ]
}
```

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-east-1:111122223333:table/Write"
            ]
        }
```

```
    ]
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Jalankan alur kerja ETL/ELT menggunakan Amazon Redshift (Lambda, API Data Amazon Redshift)

Proyek sampel ini menunjukkan cara menggunakan Step Functions dan API Data Amazon Redshift untuk menjalankan alur kerja ETL/ELT yang memuat data ke gudang data Amazon Redshift.

Dalam proyek ini, Step Functions menggunakan AWS Lambda fungsi dan Amazon Redshift Data API untuk membuat objek database yang diperlukan dan untuk menghasilkan satu set data contoh, kemudian mengeksekusi dua pekerjaan secara paralel yang melakukan pemuatan tabel dimensi, diikuti oleh tabel fakta. Setelah kedua tugas memuat dimensi berakhir dengan sukses, Step Functions mengeksekusi beban tugas untuk tabel fakta, menjalankan tugas validasi, kemudian menjeda klaster Amazon Redshift.

> **Note**
>
> Anda dapat mengubah logika ETL untuk menerima data dari sumber lain seperti Amazon S3, yang dapat menggunakan perintah SALIN untuk menyalin data dari Amazon S3 ke tabel Amazon Redshift.

Untuk informasi selengkapnya tentang integrasi layanan Amazon Redshift dan Step Functions, lihat di bawah ini:

- Menggunakan AWS Step Functions dengan layanan lain
- Menggunakan Amazon Redshift Data API
- Layanan API Data Amazon Redshift
- Membuat mesin status Step Functions yang menggunakan Lambda
- Kebijakan IAM untuk:
  - Kebijakan IAM untuk AWS Lambda
  - Mengotorisasi akses ke Amazon Redshift Data API

> **Note**
>
> Proyek sampel ini mungkin dikenakan biaya.
> Untuk AWS pengguna baru, tingkat penggunaan gratis tersedia. Pada tingkat ini, layanan akan gratis di bawah tingkat penggunaan tertentu. Untuk informasi selengkapnya tentang AWS biaya dan Tingkat Gratis, lihat AWS Step Functions harga.

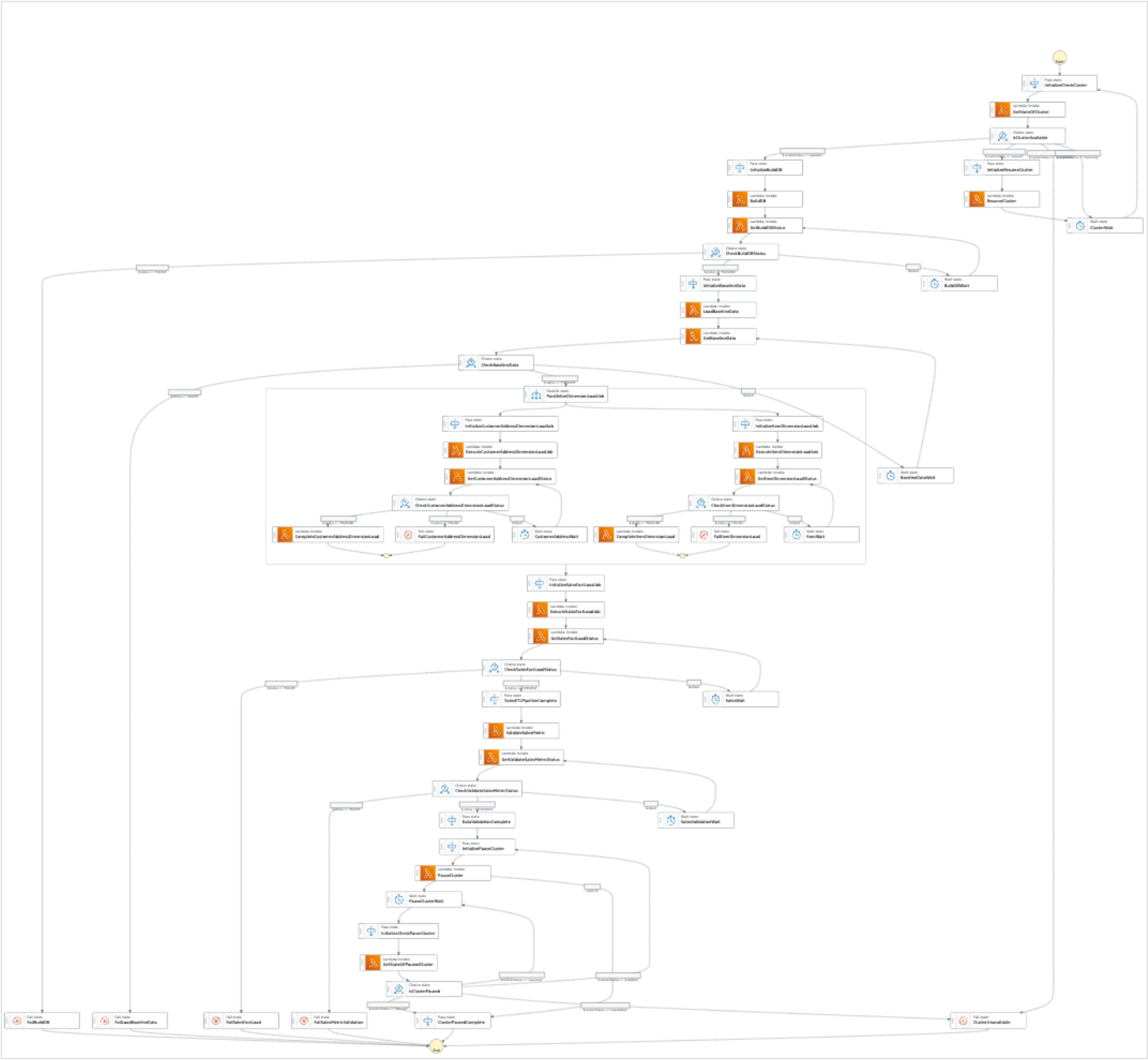
Langkah 1: Buat mesin negara dan sumber daya penyediaan

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`ETL job in Amazon Redshift`** di kotak pencarian, lalu pilih pekerjaan ETL Amazon Redshift dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

   Proyek contoh ini menyebarkan sumber daya berikut:

   - Sebuah Amazon Redshift cluster
   - Dua fungsi Lambda
   - Amazon RedshiftSkema
   - Lima Amazon Redshift tabel
   - Mesin AWS Step Functions negara
   - Peran terkait AWS Identity and Access Management (IAM).

   Gambar berikut menunjukkan grafik alur kerja untuk pekerjaan ETL dalam proyek Amazon Redshift sampel:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

     Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon

States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

> **Important**
>
> Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

  > **Tip**
  >
  > Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

  Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

  Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

  Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

  > **Important**
  >
  > Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

1. Pada halaman mesin Negara, pilih proyek sampel Anda.

2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

    1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

        > **Note**
        >
        > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

    2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

        Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

        > **Note**
        >
        > Jika proyek demo yang Anda gunakan berisi data input eksekusi yang telah diisi sebelumnya, gunakan input tersebut untuk menjalankan mesin status.

    3. Pilih Mulai Eksekusi.
    4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

        Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Contoh Kode Mesin Status

Mesin status dalam proyek sampel ini terintegrasi AWS Lambda dengan meneruskan logika ETL sebagai InputPath langsung ke sumber daya tersebut dan dieksekusi secara asinkron menggunakan Amazon Redshift Data API.

Jelajahi contoh state machine ini untuk melihat bagaimana Step Functions mengontrol AWS Lambda dan Amazon Redshift Data API.

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

```
{
"Comment": "A simple ETL workflow for loading dimension and fact tables",
  "StartAt": "InitializeCheckCluster",
  "States": {
    "InitializeCheckCluster": {
      "Type": "Pass",
      "Next": "GetStateOfCluster",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "operation": "status"
        }
      }
    },
    "GetStateOfCluster": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftOperations-AKIAIOSFODNN7EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "IsClusterAvailable",
      "InputPath": "$",
      "ResultPath": "$.clusterStatus"
    },
    "IsClusterAvailable": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "available",
          "Next": "InitializeBuildDB"
```

```
        },
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "paused",
          "Next": "InitializeResumeCluster"
        },
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "unavailable",
          "Next": "ClusterUnavailable"
        },
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "resuming",
          "Next": "ClusterWait"
        }
      ]
    },
    "ClusterWait": {
      "Type": "Wait",
      "Seconds": 720,
      "Next": "InitializeCheckCluster"
    },
    "InitializeResumeCluster": {
      "Type": "Pass",
      "Next": "ResumeCluster",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "operation": "resume"
        }
      }
    },
    "ResumeCluster": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftOperations-AKIAIOSFODNN7EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "ClusterWait",
      "InputPath": "$",
      "ResultPath": "$"
    },
    "InitializeBuildDB": {
```

```
      "Type": "Pass",
      "Next": "BuildDB",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "redshift_database": "dev",
          "redshift_user": "awsuser",
          "redshift_schema": "tpcds",
          "action": "build_database",
          "sql_statement": [
            "create schema if not exists {0} authorization {1};",
            "create table if not exists {0}.customer",
            "(c_customer_sk          int4     not null encode az64",
            ",c_customer_id          char(16) not null encode zstd",
            ",c_current_addr_sk      int4              encode az64",
            ",c_first_name           char(20)          encode zstd",
            ",c_last_name            char(30)          encode zstd",
            ",primary key (c_customer_sk)",
            ") distkey(c_customer_sk);",
            "--",
            "create table if not exists {0}.customer_address",
            "(ca_address_sk    int4     not null encode az64",
            ",ca_address_id    char(16) not null encode zstd",
            ",ca_state         char(2)           encode zstd",
            ",ca_zip           char(10)          encode zstd",
            ",ca_country       varchar(20)       encode zstd",
            ",primary key (ca_address_sk)",
            ") distkey(ca_address_sk);",
            "--",
            "create table if not exists {0}.date_dim",
            "(d_date_sk           integer  not null encode az64",
            ",d_date_id           char(16) not null encode zstd",
            ",d_date              date              encode az64",
            ",d_day_name          char(9)           encode zstd",
            ",primary key (d_date_sk)",
            ") diststyle all;",
            "--",
            "create table if not exists {0}.item",
            "(i_item_sk        int4     not null encode az64",
            ",i_item_id        char(16) not null encode zstd",
            ",i_rec_start_date date              encode az64",
            ",i_rec_end_date   date              encode az64",
            ",i_current_price  numeric(7,2)      encode az64",
            ",i_category       char(50)          encode zstd",
```

```
            ",i_product_name   char(50)          encode zstd",
            ",primary key (i_item_sk)",
            ") distkey(i_item_sk) sortkey(i_category);",
            "--",
            "create table if not exists {0}.store_sales",
            "(ss_sold_date_sk       int4",
            ",ss_item_sk            int4 not null encode az64",
            ",ss_customer_sk        int4          encode az64",
            ",ss_addr_sk            int4          encode az64",
            ",ss_store_sk           int4          encode az64",
            ",ss_ticket_number      int8 not null encode az64",
            ",ss_quantity           int4          encode az64",
            ",ss_net_paid           numeric(7,2)  encode az64",
            ",ss_net_profit         numeric(7,2)  encode az64",
            ",primary key (ss_item_sk, ss_ticket_number)",
            ") distkey(ss_item_sk) sortkey(ss_sold_date_sk);"
          ]
        }
      }
    },
    "BuildDB": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "GetBuildDBStatus",
      "InputPath": "$",
      "ResultPath": "$"
    },
    "GetBuildDBStatus": {
      "Type": "Task",
      "Next": "CheckBuildDBStatus",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "InputPath": "$",
      "ResultPath": "$.status"
    },
    "CheckBuildDBStatus": {
      "Type": "Choice",
      "Choices": [
        {
```

```
          "Variable": "$.status",
          "StringEquals": "FAILED",
          "Next": "FailBuildDB"
        },
        {
          "Variable": "$.status",
          "StringEquals": "FINISHED",
          "Next": "InitializeBaselineData"
        }
      ],
      "Default": "BuildDBWait"
    },
    "BuildDBWait": {
      "Type": "Wait",
      "Seconds": 15,
      "Next": "GetBuildDBStatus"
    },
    "FailBuildDB": {
      "Type": "Fail",
      "Cause": "Database Build Failed",
      "Error": "Error"
    },
    "InitializeBaselineData": {
      "Type": "Pass",
      "Next": "LoadBaselineData",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "redshift_database": "dev",
          "redshift_user": "awsuser",
          "redshift_schema": "tpcds",
          "action": "load_baseline_data",
          "sql_statement": [
            "begin transaction;",
            "truncate table {0}.customer;",
            "insert into {0}.customer
  (c_customer_sk,c_customer_id,c_current_addr_sk,c_first_name,c_last_name)",
            "values",
            "(7550,'AAAAAAAAOHNBAAAA',9264662,'Michelle','Deaton'),",
            "(37079,'AAAAAAAAHNAJAAAA',13971208,'Michael','Simms'),",
            "(40626,'AAAAAAAACLOJAAAA',1959255,'Susan','Ryder'),",
            "(2142876,'AAAAAAAAMJCLACAA',7644556,'Justin','Brown');",
            "analyze {0}.customer;",
            "--",
```

```
            "truncate table {0}.customer_address;",
            "insert into {0}.customer_address
(ca_address_sk,ca_address_id,ca_state,ca_zip,ca_country)",
            "values",
            "(13971208,'AAAAAAAAIAPCFNAA','NE','63451','United States'),",
            "(7644556,'AAAAAAAAMIFKEHAA','SD','58883','United States'),",
            "(9264662,'AAAAAAAAGBOFNIAA','CA','99310','United States');",
            "analyze {0}.customer_address;",
            "--",
            "truncate table {0}.item;",
            "insert into {0}.item
(i_item_sk,i_item_id,i_rec_start_date,i_rec_end_date,i_current_price,i_category,i_product_name
            "values",

"(3417,'AAAAAAAAIFNAAAAA','1997-10-27',NULL,14.29,'Electronics','ationoughtesepri
'),",
            "(9615,'AAAAAAAAOIFCAAAA','1997-10-27',NULL,9.68,'Home','antioughtcallyn
st'),",
            "(3693,'AAAAAAAAMGOAAAAA','2001-03-12',NULL,2.10,'Men','prin
stcallypri'),",

"(3630,'AAAAAAAAMCOAAAAA','2001-10-27',NULL,2.95,'Electronics','barpricallypri'),",

"(16506,'AAAAAAAAIHAEAAAA','2001-10-27',NULL,3.85,'Home','callybaranticallyought'),",

"(7866,'AAAAAAAAILOBAAAA','2001-10-27',NULL,12.60,'Jewelry','callycallyeingation');",
            "--",
            "analyze {0}.item;",
            "truncate table {0}.date_dim;",
            "insert into {0}.date_dim (d_date_sk,d_date_id,d_date,d_day_name)",
            "values",
            "(2450521,'AAAAAAAAJFEGFCAA','1997-03-13','Thursday'),",
            "(2450749,'AAAAAAAANDFGFCAA','1997-10-27','Monday'),",
            "(2451251,'AAAAAAAADDHGFCAA','1999-03-13','Saturday'),",
            "(2451252,'AAAAAAAAEDHGFCAA','1999-03-14','Sunday'),",
            "(2451981,'AAAAAAAANAKGFCAA','2001-03-12','Monday'),",
            "(2451982,'AAAAAAAAOAKGFCAA','2001-03-13','Tuesday'),",
            "(2452210,'AAAAAAAACPKGFCAA','2001-10-27','Saturday'),",
            "(2452641,'AAAAAAAABKMGFCAA','2003-01-01','Wednesday'),",
            "(2452642,'AAAAAAAACKMGFCAA','2003-01-02','Thursday');",
            "--",
            "analyze {0}.date_dim;",
            "--  commit and End transaction",
            "commit;",
```

```
            "end transaction;"
          ]
        }
      }
    },
    "LoadBaselineData": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "GetBaselineData",
      "InputPath": "$",
      "ResultPath": "$"
    },
    "GetBaselineData": {
      "Type": "Task",
      "Next": "CheckBaselineData",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "InputPath": "$",
      "ResultPath": "$.status"
    },
    "CheckBaselineData": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.status",
          "StringEquals": "FAILED",
          "Next": "FailLoadBaselineData"
        },
        {
          "Variable": "$.status",
          "StringEquals": "FINISHED",
          "Next": "ParallelizeDimensionLoadJob"
        }
      ],
      "Default": "BaselineDataWait"
    },
    "BaselineDataWait": {
      "Type": "Wait",
      "Seconds": 20,
```

```
      "Next": "GetBaselineData"
    },
    "FailLoadBaselineData": {
      "Type": "Fail",
      "Cause": "Load Baseline Data Failed",
      "Error": "Error"
    },
    "ParallelizeDimensionLoadJob": {
      "Type": "Parallel",
      "Next": "InitializeSalesFactLoadJob",
      "ResultPath": "$.status",
      "Branches": [
        {
          "StartAt": "InitializeCustomerAddressDimensionLoadJob",
          "States": {
            "InitializeCustomerAddressDimensionLoadJob": {
              "Type": "Pass",
              "Next": "ExecuteCustomerAddressDimensionLoadJob",
              "Result": {
                "input": {
                  "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
                  "redshift_database": "dev",
                  "redshift_user": "awsuser",
                  "redshift_schema": "tpcds",
                  "action": "load_customer_address",
                  "sql_statement": [
                    "begin transaction;",
                    "/* Create a staging table to hold the input data. Staging table is
created with BACKUP NO option for faster inserts and also data temporary */",
                    "drop table if exists {0}.stg_customer_address;",
                    "create table if not exists {0}.stg_customer_address",
                    "(ca_address_id    varchar(16)  encode zstd",
                    ",ca_state         varchar(2)   encode zstd",
                    ",ca_zip           varchar(10)  encode zstd",
                    ",ca_country       varchar(20)  encode zstd",
                    ")",
                    "backup no",
                    "diststyle even;",
                    "/* Ingest data from source */",
                    "insert into {0}.stg_customer_address
(ca_address_id,ca_state,ca_zip,ca_country)",
                    "values",
                    "('AAAAAAAACFBBAAAA','NE','','United States'),",
                    "('AAAAAAAAGAEFAAAA','NE','61749','United States'),",
```

```
                    "('AAAAAAAAPJKKAAAA','OK','','United States'),",
                    "('AAAAAAAAMIHGAAAA','AL','','United States');",
                    "/* Perform UPDATE for existing data with refreshed attribute
 values */",
                    "update {0}.customer_address",
                    "   set ca_state = stg_customer_address.ca_state,",
                    "       ca_zip = stg_customer_address.ca_zip,",
                    "       ca_country = stg_customer_address.ca_country",
                    "  from {0}.stg_customer_address",
                    " where customer_address.ca_address_id =
 stg_customer_address.ca_address_id;",
                    "/* Perform insert for new rows  */",
                    "insert into {0}.customer_address",
                    "(ca_address_sk",
                    ",ca_address_id",
                    ",ca_state",
                    ",ca_zip",
                    ",ca_country",
                    ")",
                    "with max_customer_address_sk as",
                    "(select max(ca_address_sk) max_ca_address_sk",
                    "from {0}.customer_address)",
                    "select row_number() over (order by
 stg_customer_address.ca_address_id) + max_customer_address_sk.max_ca_address_sk as
 ca_address_sk",
                    ",stg_customer_address.ca_address_id",
                    ",stg_customer_address.ca_state",
                    ",stg_customer_address.ca_zip",
                    ",stg_customer_address.ca_country",
                    "from {0}.stg_customer_address,",
                    "max_customer_address_sk",
                    "where stg_customer_address.ca_address_id not in (select
 customer_address.ca_address_id from {0}.customer_address);",
                    "/* Commit and End transaction */",
                    "commit;",
                    "end transaction;"
                ]
              }
            }
          },
          "ExecuteCustomerAddressDimensionLoadJob": {
            "Type": "Task",
            "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
```

```
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "Next": "GetCustomerAddressDimensionLoadStatus",
              "InputPath": "$",
              "ResultPath": "$"
            },
            "GetCustomerAddressDimensionLoadStatus": {
              "Type": "Task",
              "Next": "CheckCustomerAddressDimensionLoadStatus",
              "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "InputPath": "$",
              "ResultPath": "$.status"
            },
            "CheckCustomerAddressDimensionLoadStatus": {
              "Type": "Choice",
              "Choices": [
                {
                  "Variable": "$.status",
                  "StringEquals": "FAILED",
                  "Next": "FailCustomerAddressDimensionLoad"
                },
                {
                  "Variable": "$.status",
                  "StringEquals": "FINISHED",
                  "Next": "CompleteCustomerAddressDimensionLoad"
                }
              ],
              "Default": "CustomerAddressWait"
            },
            "CustomerAddressWait": {
              "Type": "Wait",
              "Seconds": 5,
              "Next": "GetCustomerAddressDimensionLoadStatus"
            },
            "CompleteCustomerAddressDimensionLoad": {
              "Type": "Task",
              "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "End": true
```

```
            },
            "FailCustomerAddressDimensionLoad": {
              "Type": "Fail",
              "Cause": "ETL Workflow Failed",
              "Error": "Error"
            }
          }
        },
        {
          "StartAt": "InitializeItemDimensionLoadJob",
          "States": {
            "InitializeItemDimensionLoadJob": {
              "Type": "Pass",
              "Next": "ExecuteItemDimensionLoadJob",
              "Result": {
                "input": {
                  "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
                  "redshift_database": "dev",
                  "redshift_user": "awsuser",
                  "redshift_schema": "tpcds",
                  "action": "load_item",
                  "sql_statement": [
                    "begin transaction;",
                    "/* Create a staging table to hold the input data. Staging table is
created with BACKUP NO option for faster inserts and also data temporary */",
                    "drop table if exists {0}.stg_item;",
                    "create table if not exists {0}.stg_item",
                    "(i_item_id        varchar(16) encode zstd",
                    ",i_rec_start_date date encode zstd",
                    ",i_rec_end_date   date encode zstd",
                    ",i_current_price  numeric(7,2) encode zstd",
                    ",i_category       varchar(50) encode zstd",
                    ",i_product_name   varchar(50) encode zstd",
                    ")",
                    "backup no",
                    "diststyle even;",
                    "/* Ingest data from source */",
                    "insert into {0}.stg_item",

"(i_item_id,i_rec_start_date,i_rec_end_date,i_current_price,i_category,i_product_name)",
                    "values",

"('AAAAAAAAABJBAAAA','2000-10-27',NULL,4.10,'Books','ationoughtesecally'),",
```

```
                    "('AAAAAAAAOPKBAAAA','2001-10-27',NULL,4.22,'Books','ableoughtn
 stcally'),",
                    "('AAAAAAAAHGPAAAAA','1997-10-27',NULL,29.30,'Books','priesen
 stpri'),",

 "('AAAAAAAAICMAAAAA','2001-10-27',NULL,1.93,'Books','eseoughtoughtpri'),",

 "('AAAAAAAAGPGBAAAA','2001-10-27',NULL,9.96,'Books','bareingeinganti'),",
                    "('AAAAAAAAANBEBAAAA','1997-10-27',NULL,2.25,'Music','n
 steseoughtanti'),",

 "('AAAAAAAACLAAAAAA','2001-10-27',NULL,1.71,'Home','bareingought'),",

 "('AAAAAAAAOBBDAAAA','2001-10-27',NULL,5.55,'Books','callyationantiableought');",
                    "/
*************************************************************************************
                    "** Type 2 is maintained for i_current_price column.",
                    "** Update all attributes for the item when the price is not
 changed",
                    "** Sunset existing active item record with current i_rec_end_date
 and insert a new record when the price does not match",

 "*************************************************************************************
                    "update {0}.item",
                    "   set i_category = stg_item.i_category,",
                    "       i_product_name = stg_item.i_product_name",
                    "  from {0}.stg_item",
                    " where item.i_item_id = stg_item.i_item_id",
                    "   and item.i_rec_end_date is null",
                    "   and item.i_current_price = stg_item.i_current_price;",
                    "insert into {0}.item",
                    "(i_item_sk",
                    ",i_item_id",
                    ",i_rec_start_date",
                    ",i_rec_end_date",
                    ",i_current_price",
                    ",i_category",
                    ",i_product_name",
                    ")",
                    "with max_item_sk as",
                    "(select max(i_item_sk) max_item_sk",
                    "   from {0}.item)",
                    "select row_number() over (order by stg_item.i_item_id) +
 max_item_sk as i_item_sk",
```

```
                    "      ,stg_item.i_item_id",
                    "      ,trunc(sysdate) as i_rec_start_date",
                    "      ,null as i_rec_end_date",
                    "      ,stg_item.i_current_price",
                    "      ,stg_item.i_category",
                    "      ,stg_item.i_product_name",
                    "  from {0}.stg_item, {0}.item, max_item_sk",
                    " where item.i_item_id = stg_item.i_item_id",
                    "   and item.i_rec_end_date is null",
                    "   and item.i_current_price <> stg_item.i_current_price;",
                    "/* Sunset penultimate records that were inserted as type 2 */",
                    "update {0}.item",
                    "   set i_rec_end_date = trunc(sysdate)",
                    "  from {0}.stg_item",
                    " where item.i_item_id = stg_item.i_item_id",
                    "   and item.i_rec_end_date is null",
                    "   and item.i_current_price <> stg_item.i_current_price;",
                    "/* Commit and End transaction */",
                    "commit;",
                    "end transaction;"
                  ]
                }
              }
            },
            "ExecuteItemDimensionLoadJob": {
              "Type": "Task",
              "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "Next": "GetItemDimensionLoadStatus",
              "InputPath": "$",
              "ResultPath": "$"
            },
            "GetItemDimensionLoadStatus": {
              "Type": "Task",
              "Next": "CheckItemDimensionLoadStatus",
              "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "InputPath": "$",
              "ResultPath": "$.status"
            },
```

```
            "CheckItemDimensionLoadStatus": {
              "Type": "Choice",
              "Choices": [
                {
                  "Variable": "$.status",
                  "StringEquals": "FAILED",
                  "Next": "FailItemDimensionLoad"
                },
                {
                  "Variable": "$.status",
                  "StringEquals": "FINISHED",
                  "Next": "CompleteItemDimensionLoad"
                }
              ],
              "Default": "ItemWait"
            },
            "ItemWait": {
              "Type": "Wait",
              "Seconds": 5,
              "Next": "GetItemDimensionLoadStatus"
            },
            "CompleteItemDimensionLoad": {
              "Type": "Task",
              "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "End": true
            },
            "FailItemDimensionLoad": {
              "Type": "Fail",
              "Cause": "ETL Workflow Failed",
              "Error": "Error"
            }
          }
        }
      ]
    },
    "InitializeSalesFactLoadJob": {
      "Type": "Pass",
      "Next": "ExecuteSalesFactLoadJob",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
```

```
          "redshift_database": "dev",
          "redshift_user": "awsuser",
          "redshift_schema": "tpcds",
          "snapshot_date": "2003-01-02",
          "action": "load_sales_fact",
          "sql_statement": [
            "begin transaction;",
            "/* Create a stg_store_sales staging table */",
            "drop table if exists {0}.stg_store_sales;",
            "create table {0}.stg_store_sales",
            "(sold_date             date encode zstd",
            ",i_item_id             varchar(16) encode zstd",
            ",c_customer_id         varchar(16) encode zstd",
            ",ca_address_id         varchar(16) encode zstd",
            ",ss_ticket_number      integer encode zstd",
            ",ss_quantity           integer encode zstd",
            ",ss_net_paid           numeric(7,2) encode zstd",
            ",ss_net_profit         numeric(7,2) encode zstd",
            ")",
            "backup no",
            "diststyle even;",
            "/* Ingest data from source */",
            "insert into {0}.stg_store_sales",

"(sold_date,i_item_id,c_customer_id,ca_address_id,ss_ticket_number,ss_quantity,ss_net_paid,ss_
            "values",

"('2003-01-02','AAAAAAAAIFNAAAAA','AAAAAAAAOHNBAAAA','AAAAAAAAGBOFNIAA',1403191,13,5046.37,150

"('2003-01-02','AAAAAAAAIFNAAAAA','AAAAAAAAOHNBAAAA','AAAAAAAAGBOFNIAA',1403191,13,2103.72,-12

"('2003-01-02','AAAAAAAAILOBAAAA','AAAAAAAAOHNBAAAA','AAAAAAAAGBOFNIAA',1403191,13,959.10,-130

"('2003-01-02','AAAAAAAAILOBAAAA','AAAAAAAAHNAJAAAA','AAAAAAAAIAPCFNAA',1403191,13,962.65,-475

"('2003-01-02','AAAAAAAAMCOAAAAA','AAAAAAAAHNAJAAAA','AAAAAAAAIAPCFNAA',1201746,17,111.60,-241

"('2003-01-02','AAAAAAAAMCOAAAAA','AAAAAAAAHNAJAAAA','AAAAAAAAIAPCFNAA',1201746,17,4013.02,-11

"('2003-01-02','AAAAAAAAMCOAAAAA','AAAAAAAAMJCLACAA','AAAAAAAAMIFKEHAA',1201746,17,2689.12,-55

"('2003-01-02','AAAAAAAAMGOAAAAA','AAAAAAAAMJCLACAA','AAAAAAAAMIFKEHAA',193971,18,1876.89,-556
            "/* Delete any rows from target store_sales for the input date for
idempotency */",
```

```
            "delete from {0}.store_sales where ss_sold_date_sk in (select d_date_sk
 from {0}.date_dim where d_date='{1}');",
            "/* Insert data from staging table to the target table */",
            "insert into {0}.store_sales",
            "(ss_sold_date_sk",
            ",ss_item_sk",
            ",ss_customer_sk",
            ",ss_addr_sk",
            ",ss_ticket_number",
            ",ss_quantity",
            ",ss_net_paid",
            ",ss_net_profit",
            ")",
            "select date_dim.d_date_sk ss_sold_date_sk",
            "      ,item.i_item_sk ss_item_sk",
            "      ,customer.c_customer_sk ss_customer_sk",
            "      ,customer_address.ca_address_sk ss_addr_sk",
            "      ,ss_ticket_number",
            "      ,ss_quantity",
            "      ,ss_net_paid",
            "      ,ss_net_profit",
            "  from {0}.stg_store_sales as store_sales",
            "  inner join {0}.date_dim on store_sales.sold_date = date_dim.d_date",
            "  left join {0}.item on store_sales.i_item_id = item.i_item_id and
 item.i_rec_end_date is null",
            "  left join {0}.customer on store_sales.c_customer_id =
 customer.c_customer_id",
            "  left join {0}.customer_address on store_sales.ca_address_id =
 customer_address.ca_address_id;",
            "/* Drop staging table */",
            "drop table if exists {0}.stg_store_sales;",
            "/* Commit and End transaction */",
            "commit;",
            "end transaction;"
          ]
        }
      }
    },
    "ExecuteSalesFactLoadJob": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
```

```
      "Next": "GetSalesFactLoadStatus",
      "InputPath": "$",
      "ResultPath": "$"
    },
    "GetSalesFactLoadStatus": {
      "Type": "Task",
      "Next": "CheckSalesFactLoadStatus",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "InputPath": "$",
      "ResultPath": "$.status"
    },
    "CheckSalesFactLoadStatus": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.status",
          "StringEquals": "FAILED",
          "Next": "FailSalesFactLoad"
        },
        {
          "Variable": "$.status",
          "StringEquals": "FINISHED",
          "Next": "SalesETLPipelineComplete"
        }
      ],
      "Default": "SalesWait"
    },
    "SalesWait": {
      "Type": "Wait",
      "Seconds": 5,
      "Next": "GetSalesFactLoadStatus"
    },
    "FailSalesFactLoad": {
      "Type": "Fail",
      "Cause": "ETL Workflow Failed",
      "Error": "Error"
    },
    "ClusterUnavailable": {
      "Type": "Fail",
      "Cause": "Redshift cluster is not available",
      "Error": "Error"
```

```
    },
    "SalesETLPipelineComplete": {
      "Type": "Pass",
      "Next": "ValidateSalesMetric",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "redshift_database": "dev",
          "redshift_user": "awsuser",
          "redshift_schema": "tpcds",
          "snapshot_date": "2003-01-02",
          "action": "validate_sales_metric",
          "sql_statement": [
            "select 1/count(1) from {0}.store_sales where ss_sold_date_sk in (select
 d_date_sk from {0}.date_dim where d_date='{1}')"
          ]
        }
      }
    },
    "ValidateSalesMetric": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "GetValidateSalesMetricStatus",
      "InputPath": "$",
      "ResultPath": "$"
    },
    "GetValidateSalesMetricStatus": {
      "Type": "Task",
      "Next": "CheckValidateSalesMetricStatus",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "InputPath": "$",
      "ResultPath": "$.status"
    },
    "CheckValidateSalesMetricStatus": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.status",
```

```
          "StringEquals": "FAILED",
          "Next": "FailSalesMetricValidation"
        },
        {
          "Variable": "$.status",
          "StringEquals": "FINISHED",
          "Next": "DataValidationComplete"
        }
      ],
      "Default": "SalesValidationWait"
    },
    "SalesValidationWait": {
      "Type": "Wait",
      "Seconds": 5,
      "Next": "GetValidateSalesMetricStatus"
    },
    "FailSalesMetricValidation": {
      "Type": "Fail",
      "Cause": "Data Validation Failed",
      "Error": "Error"
    },
    "DataValidationComplete": {
      "Type": "Pass",
      "Next": "InitializePauseCluster"
    },
    "InitializePauseCluster": {
      "Type": "Pass",
      "Next": "PauseCluster",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "operation": "pause"
        }
      }
    },
    "PauseCluster": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftOperations-AKIAIOSFODNN7EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "PauseClusterWait",
      "InputPath": "$",
      "ResultPath": "$.clusterStatus",
```

```
      "Catch": [
        {
          "ErrorEquals": [
            "States.ALL"
          ],
          "Next": "ClusterPausedComplete"
        }
      ]
    },
    "InitializeCheckPauseCluster": {
      "Type": "Pass",
      "Next": "GetStateOfPausedCluster",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "operation": "status"
        }
      }
    },
    "GetStateOfPausedCluster": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftOperations-AKIAIOSFODNN7EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "IsClusterPaused",
      "InputPath": "$",
      "ResultPath": "$.clusterStatus"
    },
    "IsClusterPaused": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "available",
          "Next": "InitializePauseCluster"
        },
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "paused",
          "Next": "ClusterPausedComplete"
        },
        {
          "Variable": "$.clusterStatus",
```

```
          "StringEquals": "unavailable",
          "Next": "ClusterUnavailable"
        },
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "resuming",
          "Next": "PauseClusterWait"
        }
      ]
    },
    "PauseClusterWait": {
      "Type": "Wait",
      "Seconds": 720,
      "Next": "InitializeCheckPauseCluster"
    },
    "ClusterPausedComplete": {
      "Type": "Pass",
      "End": true
    }
  }
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Contoh IAM

Contoh kebijakan AWS Identity and Access Management (IAM) yang dihasilkan oleh proyek sampel ini mencakup hak istimewa paling sedikit yang diperlukan untuk mengeksekusi mesin negara dan sumber daya terkait. Kami merekomendasikan agar Anda hanya menyertakan izin yang diperlukan dalam kebijakan IAM Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:111122223333:function:CFN36-RedshiftDataApi-
AIDACKCEVSQ6C2EXAMPLE",
```

```
                "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftOperations-AKIAIOSFODNN7EXAMPLE"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Gunakan Step Functions dan AWS Batch dengan penanganan kesalahan

Proyek sampel ini menunjukkan cara menggunakan Step Functions untuk menjalankan AWS Batch pekerjaan menggunakan mesin status dengan kemampuan penanganan kesalahan.

Dalam proyek ini, Step Functions menggunakan mesin negara untuk memanggil AWS Batch pekerjaan secara serempak. Kemudian menunggu pekerjaan berhasil atau gagal, mencoba lagi dan menangkap kesalahan ketika pekerjaan gagal, kemudian mengirim Amazon SNS topik dengan pesan tentang apakah pekerjaan itu berhasil atau gagal.

Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan

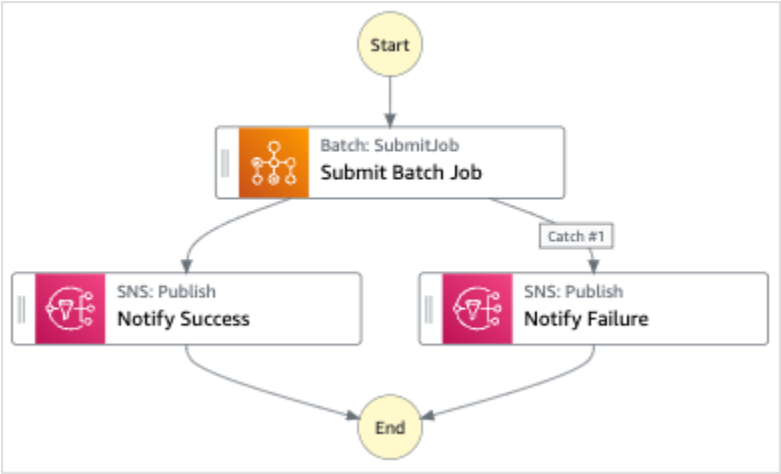
1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`Manage a batch job`** di kotak pencarian, lalu pilih Kelola pekerjaan batch dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

   Proyek contoh ini menyebarkan sumber daya berikut:

   - Sebuah AWS Batch pekerjaan
   - Topik Amazon SNS
   - Mesin AWS Step Functions negara

- Peran terkait AWS Identity and Access Management (IAM)

Gambar berikut menunjukkan grafik alur kerja untuk Mengelola proyek sampel pekerjaan batch:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

- Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

  Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

  > **Important**
  >
  > Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

> **Tip**
>
> Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

> **Important**
>
> Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

1. Pada halaman mesin Negara, pilih proyek sampel Anda.
2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

    1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

    > **Note**
    >
    > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini

tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

   Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

   > **Note**
   >
   > Jika proyek demo yang Anda gunakan berisi data input eksekusi yang telah diisi sebelumnya, gunakan input tersebut untuk menjalankan mesin status.

3. Pilih Mulai Eksekusi.
4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

   Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Contoh Kode Mesin Status

Mesin status dalam proyek sampel ini terintegrasi dengan AWS Batch dan Amazon SNS dengan meneruskan parameter langsung ke sumber daya tersebut.

Jelajahi contoh mesin status ini untuk melihat bagaimana Step Functions mengontrol AWS Batch dan Amazon SNS dengan menghubungkan ke Amazon Resource Name (ARN) di `Resource` bidang, dan dengan meneruskan `Parameters` ke API layanan.

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

```
{
  "Comment": "An example of the Amazon States Language for notification on an AWS Batch
 job completion",
```

```
  "StartAt": "Submit Batch Job",
  "TimeoutSeconds": 3600,
  "States": {
    "Submit Batch Job": {
      "Type": "Task",
      "Resource": "arn:aws:states:::batch:submitJob.sync",
      "Parameters": {
        "JobName": "BatchJobNotification",
        "JobQueue": "arn:aws:batch:us-west-2:123456789012:job-queue/
BatchJobQueue-123456789abcdef",
        "JobDefinition": "arn:aws:batch:us-west-2:123456789012:job-definition/
BatchJobDefinition-123456789abcdef:1"
      },
      "Next": "Notify Success",
      "Retry": [
        {
          "ErrorEquals": [
            "States.ALL"
          ],
          "IntervalSeconds": 30,
          "MaxAttempts": 2,
          "BackoffRate": 1.5
        }
      ],
      "Catch": [
        {
          "ErrorEquals": [ "States.ALL" ],
          "Next": "Notify Failure"
        }
      ]
    },
    "Notify Success": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
        "Message": "Batch job submitted through Step Functions succeeded",
        "TopicArn": "arn:aws:sns:us-west-2:123456789012:StepFunctionsSample-
BatchJobManagement12345678-9abc-def0-1234-567890abcdef-SNSTopic-A2B3C4D5E6F7G"
      },
      "End": true
    },
    "Notify Failure": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
```

```
      "Parameters": {
        "Message": "Batch job submitted through Step Functions failed",
        "TopicArn": "arn:aws:sns:us-west-2:123456789012:StepFunctionsSample-
BatchJobManagement12345678-9abc-def0-1234-567890abcdef-SNSTopic-A2B3C4D5E6F7G"
      },
      "End": true
    }
  }
}
```

Contoh IAM

Contoh kebijakan AWS Identity and Access Management (IAM) yang dihasilkan oleh proyek sampel ini mencakup hak istimewa paling sedikit yang diperlukan untuk mengeksekusi mesin negara dan sumber daya terkait. Kami merekomendasikan agar Anda hanya menyertakan izin yang diperlukan dalam kebijakan IAM Anda.

Example **`BatchJobNotificationAccessPolicy`**

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:us-west-2:123456789012:StepFunctionsSample-
BatchJobManagement12345678-9abc-def0-1234-567890abcdef-SNSTopic-A2B3C4D5E6F7G"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "batch:SubmitJob",
                "batch:DescribeJobs",
                "batch:TerminateJob"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
```

```
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:us-west-2:123456789012:rule/
StepFunctionsGetEventsForBatchJobsRule"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Menggemari AWS Batch pekerjaan

Proyek contoh ini mendemonstrasikan cara menggunakan Map status Step Functions untuk menyebarkan AWS Batch pekerjaan.

Dalam proyek ini, Step Functions menggunakan mesin status untuk memanggil fungsi Lambda untuk melakukan pra-pemrosesan sederhana, lalu memanggil beberapa AWS Batch pekerjaan secara paralel menggunakan status. Map

Langkah 1: Buat mesin negara bagian dan sumber daya penyediaan

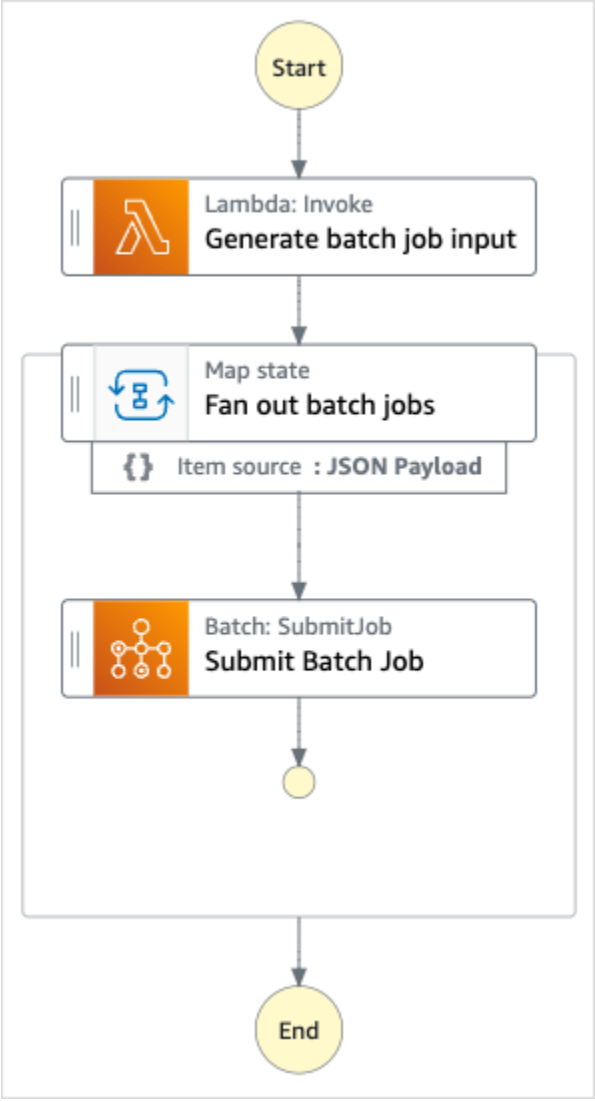
1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`Fan out a batch job`** di kotak pencarian, lalu pilih Fan out a batch job dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

   Proyek contoh ini menyebarkan sumber daya berikut:

   - Sebuah Lambda fungsi

- Antrian AWS Batch pekerjaan
- Mesin AWS Step Functions negara
- Peran terkait AWS Identity and Access Management (IAM)

Gambar berikut menunjukkan grafik alur kerja untuk proyek sampel pekerjaan batch Fan out a batch:



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

> **Important**
>
> Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

> **Tip**
>
> Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

> **Important**
>
> Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

1. Pada halaman mesin Negara, pilih proyek sampel Anda.
2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

    1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

        > **Note**
        >
        > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

    2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

        Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

        > **Note**
        >
        > Jika proyek demo yang Anda gunakan berisi data input eksekusi yang telah diisi sebelumnya, gunakan input tersebut untuk menjalankan mesin status.

    3. Pilih Mulai Eksekusi.
    4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

        Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Contoh Kode Mesin Status

Mesin status dalam proyek sampel ini terintegrasi dengan AWS Batch dan Amazon SNS dengan meneruskan parameter langsung ke sumber daya tersebut.

Jelajahi contoh mesin status ini untuk melihat bagaimana Step Functions mengontrol AWS Batch dan Amazon SNS dengan menghubungkan ke Amazon Resource Name (ARN) di `Resource` bidang, dan dengan meneruskan `Parameters` ke API layanan.

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

```
{
  "Comment": "An example of the Amazon States Language for fanning out AWS Batch job",
  "StartAt": "Generate batch job input",
  "TimeoutSeconds": 3600,
  "States": {
    "Generate batch job input": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "OutputPath": "$.Payload",
      "Parameters": {
        "FunctionName": "<GENERATE_BATCH_JOB_INPUT_LAMBDA_FUNCTION_NAME>"
      },
      "Next": "Fan out batch jobs"
    },
    "Fan out batch jobs": {
      "Comment": "Start multiple executions of batch job depending on pre-processed
 data",
      "Type": "Map",
      "End": true,
      "ItemsPath": "$",
      "Parameters": {
        "BatchNumber.$": "$$.Map.Item.Value"
      },
      "Iterator": {
        "StartAt": "Submit Batch Job",
        "States": {
          "Submit Batch Job": {
            "Type": "Task",
            "Resource": "arn:aws:states:::batch:submitJob.sync",
            "Parameters": {
              "JobName": "BatchJobFanOut",
```

```
              "JobQueue": "<BATCH_QUEUE_ARN>",
              "JobDefinition": "<BATCH_JOB_DEFINITION_ARN>"
            },
            "End": true
          }
        }
      }
    }
  }
}
```

Contoh IAM

Contoh kebijakan AWS Identity and Access Management (IAM) yang dihasilkan oleh proyek sampel ini mencakup hak istimewa paling sedikit yang diperlukan untuk mengeksekusi mesin negara dan sumber daya terkait. Kami merekomendasikan agar Anda hanya menyertakan izin yang diperlukan dalam kebijakan IAM Anda.

Example **`BatchJobFanOutAccessPolicy`**

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "batch:SubmitJob",
                "batch:DescribeJobs",
                "batch:TerminateJob"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:us-west-2:123456789012:rule/
StepFunctionsGetEventsForBatchJobsRule"
            ],
```

```
            "Effect": "Allow"
        }
    ]
}
```

Example **`InvokeGenerateBatchJobMapLambdaPolicy`**

```
{
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:us-
west-2:123456789012:function:StepFunctionsSample-BatchJobFa-
GenerateBatchJobMap-444455556666",
            "Effect": "Allow"
        }
    ]
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

AWS Batch dengan Lambda

Proyek sampel ini menunjukkan cara menggunakan Step Functions untuk melakukan pra-proses data dengan AWS Lambda fungsi dan kemudian mengatur AWS Batch pekerjaan.

Dalam proyek ini, Step Functions menggunakan mesin status untuk memanggil fungsi Lambda untuk melakukan pra-pemrosesan sederhana sebelum pekerjaan AWS Batch dikirimkan. Beberapa pekerjaan dapat dipanggil tergantung pada hasil atau keberhasilan yang sebelumnya.

Langkah 1: Buat Mesin Negara dan Sumber Daya Penyediaan

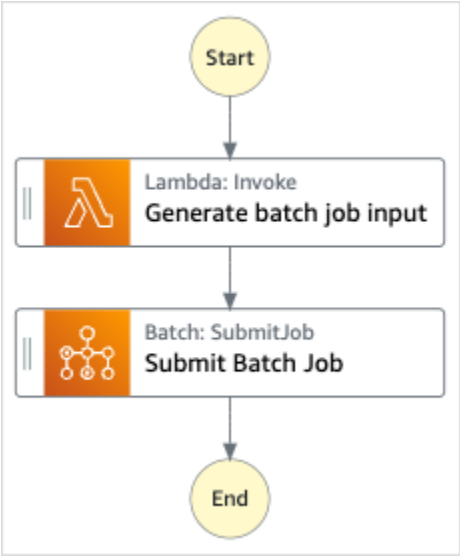
1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`Batch job with Lambda`** di kotak pencarian, lalu pilih Pekerjaan batch dengan Lambda dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.

4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

   Proyek contoh ini menyebarkan sumber daya berikut:

   - Sebuah Lambda fungsi
   - Tugas AWS Batch
   - Mesin AWS Step Functions negara
   - Peran terkait AWS Identity and Access Management (IAM)

   Gambar berikut menunjukkan grafik alur kerja untuk pekerjaan Batch dengan proyek Lambda sampel:

5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

   - Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

     Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon

States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

> **Important**
>
> Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

  > **Tip**
  >
  > Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

  Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

  Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

  Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

  > **Important**
  >
  > Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

1. Pada halaman mesin Negara, pilih proyek sampel Anda.

2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

    1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions menghasilkan nama eksekusi unik secara otomatis.

        > **Note**
        >
        > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

    2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

        Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

        > **Note**
        >
        > Jika proyek demo yang Anda gunakan berisi data input eksekusi yang telah diisi sebelumnya, gunakan input tersebut untuk menjalankan mesin status.

    3. Pilih Mulai Eksekusi.
    4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

        Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output, dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Contoh Kode Mesin Status

Mesin status dalam proyek sampel ini terintegrasi dengan AWS Batch dan Amazon SNS dengan meneruskan parameter langsung ke sumber daya tersebut.

Jelajahi contoh mesin status ini untuk melihat bagaimana Step Functions mengontrol AWS Batch dan Amazon SNS dengan menghubungkan ke Amazon Resource Name (ARN) di `Resource` bidang, dan dengan meneruskan `Parameters` ke API layanan.

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihat[Menggunakan AWS Step Functions dengan layanan lain](#).

```
{
  "Comment": "An example of the Amazon States Language for using batch job with pre-
processing lambda",
  "StartAt": "Generate batch job input",
  "TimeoutSeconds": 3600,
  "States": {
    "Generate batch job input": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "OutputPath": "$.batch_input",
      "Parameters": {
        "FunctionName": "<GENERATE_BATCH_JOB_INPUT_LAMBDA_FUNCTION_NAME>"
      },
      "Next": "Submit Batch Job"
    },
    "Submit Batch Job": {
      "Type": "Task",
      "Resource": "arn:aws:states:::batch:submitJob.sync",
      "Parameters": {
        "JobName": "BatchJobFanOut",
        "JobQueue": "<BATCH_QUEUE_ARN>",
        "JobDefinition": "<BATCH_JOB_DEFINITION_ARN>",
        "Parameters.$": "$.batch_input"
      },
      "End": true
    }
  }
}
```

Contoh IAM

Contoh kebijakan AWS Identity and Access Management (IAM) yang dihasilkan oleh proyek sampel ini mencakup hak istimewa paling sedikit yang diperlukan untuk mengeksekusi mesin negara dan sumber daya terkait. Kami merekomendasikan agar Anda hanya menyertakan izin yang diperlukan dalam kebijakan IAM Anda.

Example **`BatchJobWithLambdaAccessPolicy`**

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:us-west-2:123456789012:ManageBatchJob-SNSTopic-
JHLYYG7AZPZI"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "batch:SubmitJob",
                "batch:DescribeJobs",
                "batch:TerminateJob"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:us-west-2:123456789012:rule/
StepFunctionsGetEventsForBatchJobsRule"
            ],
            "Effect": "Allow"
        }
```

```
    ]
}
```

Example **`InvokeGenerateBatchJobMapLambdaPolicy`**

```
{
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:us-
west-2:123456789012:function:StepFunctionsSample-BatchWithL-
GenerateBatchJobMap-444455556666",
            "Effect": "Allow"
        }
    ]
}
```

Untuk informasi tentang cara mengonfigurasi IAM saat menggunakan Step Functions dengan AWS layanan lain, lihatKebijakan IAM untuk layanan terintegrasi.

Lakukan AI prompt-chaining dengan Amazon Bedrock

Proyek sampel ini menunjukkan bagaimana Anda dapat berintegrasi dengan Amazon Bedrock untuk melakukan prompt chaining AI. Contoh proyek ini menunjukkan bagaimana Anda dapat membuat chatbot berkualitas tinggi menggunakan. Amazon Bedrock Proyek menggabungkan beberapa petunjuk dan menyelesaikannya dalam urutan di mana mereka disediakan. Rantai petunjuk ini menambah kemampuan model bahasa yang digunakan untuk memberikan respons yang sangat dikuratori.

Proyek sampel ini membuat mesin status, AWS sumber daya pendukung, dan mengonfigurasi izin IAM terkait. Jelajahi proyek sampel ini untuk mempelajari tentang menggunakan integrasi layanan yang Amazon Bedrock dioptimalkan dengan mesin Step Functions negara, atau gunakan sebagai titik awal untuk proyek Anda sendiri.

Topik

- Templat AWS CloudFormation dan sumber daya tambahan
- Prasyarat

- Langkah 1: Buat mesin negara dan sumber daya penyediaan
- Langkah 2: Jalankan mesin negara

Templat AWS CloudFormation dan sumber daya tambahan

Anda menggunakan CloudFormation template untuk menyebarkan proyek sampel ini. Template ini membuat sumber daya berikut di Akun AWS:

- Mesin Step Functions negara.
- Peran eksekusi untuk mesin negara. Peran ini memberikan izin yang dibutuhkan mesin status Anda untuk mengakses sumber lain Layanan AWS dan sumber daya seperti tindakan. Amazon Bedrock InvokeModel

Prasyarat

Proyek sampel ini menggunakan Cohere Command large language model (LLM). Agar berhasil menjalankan proyek sampel ini, Anda harus menambahkan akses ke LLM ini dari Amazon Bedrock konsol. Untuk menambahkan akses model, lakukan hal berikut:

1. Buka konsol Amazon Bedrock.
2. Pada panel navigasi, pilih Akses model.
3. Pilih Kelola akses model.
4. Pilih kotak centang di sebelah Cohere.
5. Pilih Minta akses. Status Access untuk model Cohere ditampilkan sebagai Access diberikan.

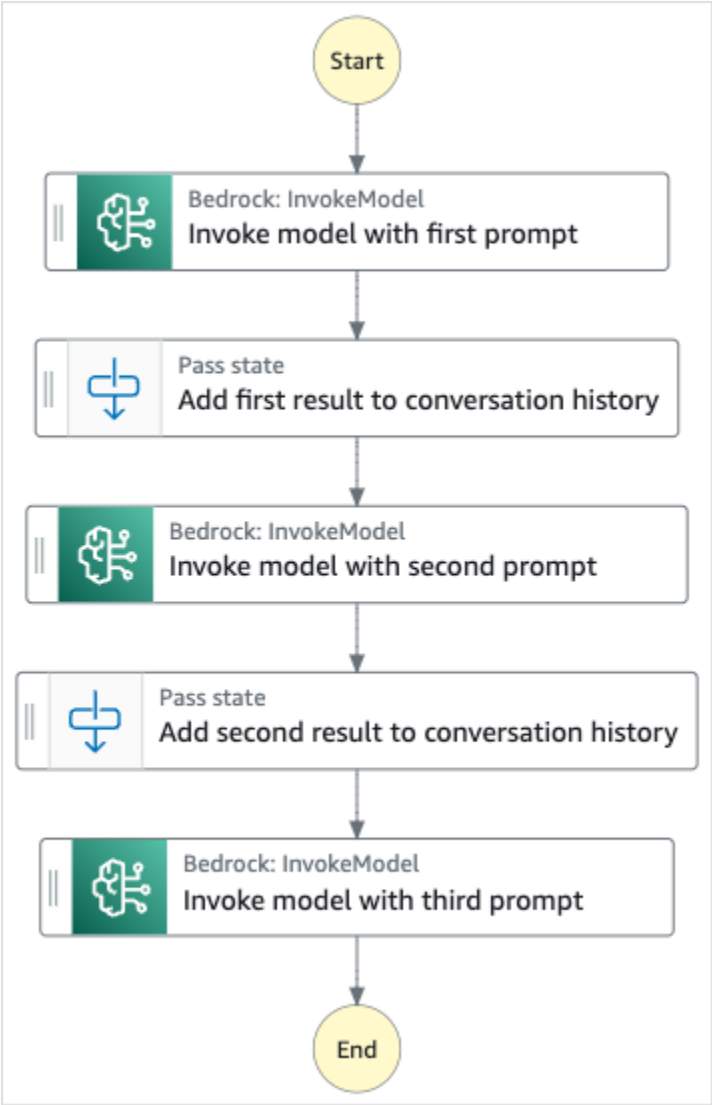
Langkah 1: Buat mesin negara dan sumber daya penyediaan

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Ketik **`bedrock`** di kotak pencarian, lalu pilih Lakukan AI prompt-chaining dengan Bedrock dari hasil pencarian yang dikembalikan.
3. Pilih Next untuk melanjutkan.
4. Step Functions mencantumkan yang Layanan AWS digunakan dalam proyek sampel yang Anda pilih. Ini juga menunjukkan grafik alur kerja untuk proyek sampel. Menyebarkan proyek ini ke Anda Akun AWS atau menggunakannya sebagai titik awal untuk membangun proyek Anda sendiri. Berdasarkan cara Anda ingin melanjutkan, pilih Jalankan demo atau Bangun di atasnya.

Proyek contoh ini menyebarkan sumber daya berikut:

- Mesin AWS Step Functions negara
- Peran terkait AWS Identity and Access Management (IAM)

Gambar berikut menunjukkan grafik alur kerja untuk project prompt Perform AI with sample: Bedrock



5. Pilih Gunakan templat untuk melanjutkan pilihan Anda.
6. Lakukan salah satu hal berikut:

- Jika Anda memilih Build on it, Step Functions akan membuat prototipe alur kerja untuk proyek sampel yang Anda pilih. Step Functions tidak menyebarkan sumber daya yang tercantum dalam definisi alur kerja.

  Di Workflow StudioMode desain, seret dan lepas status dari Peramban status untuk terus membangun prototipe alur kerja Anda. Atau beralih ke Mode kode yang menyediakan editor kode terintegrasi yang mirip dengan VS Code untuk memperbarui definisi Amazon States Language (ASL) mesin status Anda dalam konsol Step Functions. Untuk informasi selengkapnya tentang penggunaan Workflow Studio untuk membangun mesin status Anda, lihatMenggunakan Workflow Studio.

  > **Important**
  >
  > Ingatlah untuk memperbarui nama sumber daya Amazon (ARN) placeholder untuk sumber daya yang digunakan dalam proyek sampel sebelum Anda menjalankan alur kerja.

- Jika Anda memilih Jalankan demo, Step Functions akan membuat proyek sampel hanya-baca yang menggunakan AWS CloudFormation templat untuk menyebarkan AWS sumber daya yang tercantum dalam templat tersebut ke templat Anda. Akun AWS

  > **Tip**
  >
  > Untuk melihat definisi mesin status dari proyek sampel, pilih Kode.

  Saat Anda siap, pilih Deploy dan jalankan untuk menyebarkan proyek sampel dan membuat sumber daya.

  Tindakan ini dapat memakan waktu hingga 10 menit untuk membuat sumber daya dan izin IAM terkait. Saat sumber daya Anda sedang digunakan, Anda dapat membuka tautan CloudFormation Stack ID untuk melihat sumber daya mana yang sedang disediakan.

  Setelah semua sumber daya dalam proyek sampel dibuat, Anda dapat melihat proyek sampel baru yang tercantum di halaman mesin Negara.

> **Important**
>
> Biaya standar mungkin berlaku untuk setiap layanan yang digunakan dalam CloudFormation templat.

Langkah 2: Jalankan mesin negara

1. Pada halaman mesin Negara, pilih proyek sampel Anda.
2. Pada halaman proyek sampel, pilih Mulai eksekusi.
3. Dalam kotak dialog Mulai eksekusi, lakukan hal berikut:

    1. (Opsional) Untuk mengidentifikasi eksekusi Anda, Anda dapat menentukan nama untuk itu di Nama kotak. Secara default, Step Functions secara otomatis menghasilkan nama eksekusi yang unik.

        > **Note**
        >
        > Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, aktivitas, dan label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

    2. (Opsional) Dalam kotak Input, masukkan nilai input dalam format JSON untuk menjalankan alur kerja Anda.

        Jika Anda memilih untuk Menjalankan demo, Anda tidak perlu memberikan input eksekusi apa pun.

    3. Pilih Mulai Eksekusi.
    4. Konsol Step Functions mengarahkan Anda ke halaman yang berjudul dengan ID eksekusi Anda. Halaman ini dikenal sebagai halaman Detail Eksekusi. Di halaman ini, Anda dapat meninjau hasil eksekusi saat eksekusi berlangsung atau setelah selesai.

        Untuk meninjau hasil eksekusi, pilih status individual pada tampilan Grafik, lalu pilih tab individual di Detail langkah panel untuk melihat detail setiap status termasuk input, output,

dan definisi masing-masing. Untuk detail tentang informasi eksekusi yang dapat Anda lihat di halaman Rincian Eksekusi, lihatHalaman Detail Eksekusi - Ikhtisar antarmuka.

Kuota

AWS Step Functions menempatkan kuota pada ukuran parameter mesin status tertentu, seperti jumlah tindakan API selama periode waktu tertentu atau jumlah mesin status yang dapat Anda tentukan. Meskipun kuota ini dirancang untuk mencegah mesin status melakukan kesalahan konfigurasi dengan mengonsumsi semua sumber daya dari sistem, sebagian besar bukan merupakan kuota keras.

Untuk meminta peningkatan kuota layanan, Anda dapat melakukan salah satu hal berikut:

- Gunakan konsol Service Quotas di https://console.aws.amazon.com/servicequotas/home. Untuk informasi tentang meminta peningkatan kuota menggunakan konsol Service Quotas, lihat Meminta peningkatan kuota dalam Panduan Pengguna Service Quotas.
- Gunakan halaman Support Center AWS Management Console untuk meminta peningkatan kuota sumber daya yang disediakan oleh AWS Step Functions per wilayah. Untuk informasi selengkapnya, lihat kuota layanan AWS di Referensi Umum AWS.

> **Note**
>
> Jika tahap tertentu eksekusi mesin status atau aktivitas eksekusi terlalu lama, Anda dapat mengonfigurasi batas waktu mesin status untuk menghasilkan peristiwa waktu habis.

Topik

- Kuota umum
- Kuota yang terkait dengan akun
- Kuota yang terkait dengan HTTP Task
- Kuota terkait throttling status
- Kuota terkait throttling tindakan API
- Kuota yang berkaitan dengan eksekusi mesin status
- Kuota yang berkaitan dengan eksekusi tugas
- Kuota yang terkait dengan versi dan alias
- Pembatasan terkait penandaan

Kuota umum

Kuota	Deskripsi
Nama dalam Step Functions	Nama mesin negara, eksekusi, dan tugas aktivitas tidak boleh melebihi 80 karakter panjangnya. Nama-nama ini harus unik untuk akun dan AWS Wilayah Anda, dan tidak boleh mengandung salah satu dari yang berikut: • Spasi putih • Karakter wildcard (`? *`) • Karakter tanda kurung (`< > { } [ ]`) • Karakter khusus (`" # % \ ^ | ~ ` $ & , ; : /`) • Karakter kontrol (`\\u0000` - `\\u001f` atau `\ \u007f` - `\\u009f`). Jika mesin status Anda bertipe Express, Anda dapat memberikan nama yang sama untuk beberapa eksekusi mesin status. Step Functions menghasilkan ARN eksekusi unik untuk setiap eksekusi mesin status Express, bahkan jika beberapa eksekusi memiliki nama yang sama. Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas, serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.

Kuota yang terkait dengan akun

Sumber daya	Kuota default	Dapat ditingkatkan hingga
Jumlah maksimum dari mesin status yang terdaftar	10.000	25.000
Jumlah maksimum aktivitas yang terdaftar	10.000	15.000
Ukuran permintaan maksimum	1 MB per permintaan. Ini adalah ukuran data total per permintaan API Step Functions, termasuk header permintaan dan semua data permintaan terkait lainnya.	Kuota keras
Eksekusi terbuka maksimal per akun	1.000.000 eksekusi untuk masing-masing Akun AWS . Wilayah AWS Melebihi ini akan menyebabkan kesalahan `ExecutionLimitExceeded` . Ini tidak berlaku untuk Alur Kerja Ekspres.	Juta.
Jumlah maksimum Map Runs yang terbuka Map Run terbuka adalah Map Run yang telah dimulai, tetapi belum selesai. Peta Terjadwal Runs menunggu di MapRunStartedacara hingga jumlah total Map Runs yang terbuka kurang dari kuota default 1000.	1000 Kuota ini berlaku untuk status Peta Terdistribusi.	Kuota keras

Sumber daya	Kuota default	Dapat ditingkatkan hingga
Maksimum redrivesdari Map Run.	1000 Kuota ini berlaku untuk status Peta Terdistribusi.	Kuota keras

Kuota yang terkait dengan HTTP Task

Tugas HTTP dibatasi menggunakan skema token bucket untuk mempertahankan bandwidth Step Functions layanan. Tabel berikut mencantumkan ukuran bucket dan laju isi ulang untuk Tugas HTTP.

Sumber daya	Ukuran bucket	Tingkat isi ulang per detik
Tugas HTTP	300	300

Tabel berikut mencantumkan kuota untuk durasi HTTP Task.

Sumber daya	Kuota bawaan
Durasi tugas HTTP Durasi Tugas HTTP mengacu pada waktu yang dibutuhkan oleh Tugas HTTP untuk mengirim permintaan HTTP dan menerima respons.	60 detik Ini adalah kuota keras yang tidak dapat diubah.

Kuota terkait throttling status

Transisi status Step Functions dibatasi menggunakan skema token bucket untuk mempertahankan bandwidth layanan. Alur kerja standar dan alur kerja Express memiliki status transisi throttling yang berbeda. Kuota alur kerja standar adalah kuota lunak dan dapat ditingkatkan.

> **Note**
>
> Pelambatan pada metrik `StateTransition` layanan dilaporkan seperti di `ExecutionThrottled` Amazon. CloudWatch Untuk informasi lebih lanjut, lihat `ExecutionThrottled` CloudWatch metrik.

	Standard		Express	
Metrik layanan	Ukuran bucket	Tingkat isi ulang per detik	Ukuran bucket	Tingkat isi ulang per detik
`StateTransition` — Di US East (N. Virginia), US West (Oregon), dan Europe (Ireland)	5.000	5.000	Tidak terbatas.	Tidak terbatas.
`StateTransition` — Semua daerah lainnya	800	800	Tidak terbatas.	Tidak terbatas.

Kuota terkait throttling tindakan API

Beberapa tindakan API Step Functions dibatasi menggunakan skema bucket token untuk mempertahankan bandwidth layanan. Kuota ini adalah kuota lunak dan dapat ditingkatkan.

> **Note**
>
> Kuota pembatasan adalah per akun, per Wilayah. AWS AWS Step Functions dapat meningkatkan ukuran ember dan laju isi ulang kapan saja.

	Standard		Express	
Nama API	Ukuran bucket	Tingkat isi ulang per detik	Ukuran bucket	Tingkat isi ulang per detik
`StartExecution` — Di US East (N. Virginia), US West (Oregon), dan Europe (Ireland)	1.300	300	6.000	6.000
`StartExecution` — Semua daerah lainnya	800	150	6.000	6.000

Kuota yang terkait dengan API TestState

Tabel berikut mencantumkan kuota yang tersedia untuk TestState API.

Nama API	Kuota	Dapat ditingkatkan hingga
TestState	1 transaksi per detik (TPS)	Kuota keras

Kuota lainnya

Kuota ini adalah kuota lunak dan dapat ditingkatkan.

	In US East (N. Virginia), US West (Oregon), and Europe (Ireland)		All other regions	
Nama API	Ukuran bucket	Tingkat isi ulang per detik	Ukuran bucket	Tingkat isi ulang per detik
`CreateActivity`	100	1	100	1
`CreateStateMachine`	100	1	100	1
`DeleteActivity`	100	1	100	1
`DeleteStateMachine`	100	1	100	1
`DescribeActivity`	200	1	200	1
`DescribeExecution`	300	15	250	10
`DescribeStateMachine`	200	20	200	20
`DescribeStateMachineForExecution`	200	1	200	1
`GetActivityTask`	3.000	500	1.500	300
`GetExecutionHistory`	400	20	400	20

	In US East (N. Virginia), US West (Oregon), and Europe (Ireland)		All other regions	
Nama API	Ukuran bucket	Tingkat isi ulang per detik	Ukuran bucket	Tingkat isi ulang per detik
`ListActivities`	100	10	100	5
`ListExecutions`	200	5	100	2
`ListStateMachines`	100	5	100	5
`ListTagsForResource`	100	1	100	1
`SendTaskFailure`	3.000	500	1.500	300
`SendTaskHeartbeat`	3.000	500	1.500	300
`SendTaskSuccess`	3.000	500	1.500	300

	In US East (N. Virginia), US West (Oregon), and Europe (Ireland)		All other regions	
Nama API	Ukuran bucket	Tingkat isi ulang per detik	Ukuran bucket	Tingkat isi ulang per detik
StartSync Execution	Panggilan API eksekusi Express sinkron tidak berkontri busi pada batas kapasitas akun yang ada. Step Functions menyediakan kapasitas sesuai permintaan dan secara otomatis menskalak an dengan beban kerja yang berkelanj utan. Lonjakan beban kerja dapat di-throtling hingga kapasitas tersedia. Jika Anda mengalami pelambatan, coba lagi setelah beberapa waktu. Untuk informasi tentang alur kerja Synchrono us Express,			

	In US East (N. Virginia), US West (Oregon), and Europe (Ireland)		All other regions	
Nama API	Ukuran bucket	Tingkat isi ulang per detik	Ukuran bucket	Tingkat isi ulang per detik
	lihat. Alur kerja Ekspres Sinkron dan Tidak Sinkron			
`StopExecution`	1.000	200	500	25
`TagResource`	200	1	200	1
`UntagResource`	200	1	200	1
`UpdateStateMachine`	100	1	100	1

Kuota yang berkaitan dengan eksekusi mesin status

Tabel berikut menjelaskan kuota yang terkait dengan eksekusi mesin status. Kuota eksekusi mesin status adalah kuota keras yang tidak dapat diubah, kecuali untuk kuota waktu retensi riwayat Eksekusi.

Kuota	Standar	Express
Waktu eksekusi maksimum	1 tahun. Jika eksekusi berjalan lebih dari maksimum 1 tahun, itu akan gagal dengan `States.Timeout` kesalahan dan memancarkan metrik. `ExecutionsTimedOut` CloudWatch	5 menit. Jika eksekusi berjalan lebih dari maksimum 5 menit, itu akan gagal dengan `States.Timeout` kesalahan dan memancarkan metrik. `ExecutionsTimedOut` CloudWatch

Kuota	Standar	Express
Ukuran riwayat eksekusi maksimum	25.000 peristiwa dalam satu riwayat eksekusi mesin negara. Jika riwayat eksekusi mencapai kuota ini, eksekusi akan gagal. Untuk menghindari hal ini, lihat Hindari mencapai kuota sejarah.	Tidak terbatas.
Waktu eksekusi maksimum	1 tahun (dibatasi oleh waktu eksekusi maksimum).	5 menit (dibatasi oleh waktu eksekusi maksimum).

Kuota	Standar	Express
Waktu retensi riwayat eksekusi	90 hari setelah eksekusi ditutup. Setelah waktu ini, Anda tidak lagi dapat mengambil atau melihat riwayat eksekusi. Tidak ada kuota lebih lanjut untuk jumlah eksekusi tertutup yang dipertahankan Step Functions. Untuk memenuhi persyaratan kepatuhan, organisasi, atau peraturan, Anda dapat mengurangi periode retensi riwayat eksekusi menjadi 30 hari dengan mengirimkan permintaan kuota. Untuk melakukan ini, gunakan AWS Support Center Console dan buat kasus baru. Perubahan untuk mengurangi periode retensi menjadi 30 hari berlaku untuk setiap akun di Wilayah.	Untuk melihat riwayat eksekusi, CloudWatch pencatatan Log Amazon harus dikonfigurasi. Untuk informasi selengkapnya, lihat Logging menggunakanCloudWatchLog.
redrivablePeriode eksekusi Redrivableperiode mengacu pada waktu di mana Anda dapat redriveeksekusi Alur Kerja Standar tertentu. Periode ini dimulai dari hari mesin negara menyelesaikan pelaksanaannya.	14 hari. Kuota keras ini berlaku untuk status Peta Terdistribusi.	Redrivesaat ini tidak didukung untuk alur kerja Express.

Kuota yang berkaitan dengan eksekusi tugas

Tabel berikut menjelaskan kuota terkait eksekusi tugas, dan kuota keras yang tidak dapat diubah.

Kuota	Standar	Express
Waktu eksekusi tugas maksimum	1 tahun (dibatasi oleh waktu eksekusi maksimal)	5 menit (dibatasi oleh waktu eksekusi maksimal)
Waktu maksimum bagi Step Functions menyimpan tugas dalam antrean	1 tahun (dibatasi oleh waktu eksekusi maksimal)	5 menit (dibatasi oleh waktu eksekusi maksimal)
Poling aktivitas maksimum per Amazon Resource Name (ARN)	1.000 poller memanggil `GetActivityTask` per ARN. Melebihi kuota ini berakibat kesalahan berikut : "Jumlah maksimum pekerja serentak untuk tugas kegiatan telah tercapai."	Tidak berlaku untuk alur kerja Express.
Ukuran input atau output maksimum untuk tugas, sttaus, atau eksekusi	256 KB data sebagai string yang dikodekan UTF-8. Kuota ini mempengaruhi tugas (aktivitas, fungsi Lambda, atau layanan terpadu), output status atau eksekusi, dan data input saat penjadwalan tugas, memasuki status, atau memulai eksekusi.	256 KB data sebagai string yang dikodekan UTF-8. Kuota ini mempengaruhi tugas (aktivitas, fungsi Lambda, atau layanan terpadu), output status atau eksekusi, dan data input saat penjadwalan tugas, memasuki status, atau memulai eksekusi.

Kuota yang terkait dengan versi dan alias

Sumber daya	Kuota bawaan
Jumlah maksimum versi mesin negara yang dipublikasikan	1000 untuk setiap mesin negara. Untuk meminta peningkatan batas lunak ini, gunakan halaman Support Center di halaman AWS Management Console.
Jumlah maksimum alias mesin negara	100 untuk setiap mesin negara. Untuk meminta peningkatan batas lunak ini, gunakan halaman Support Center di halaman AWS Management Console.

Pembatasan terkait penandaan

Waspadalah terhadap pembatasan ini ketika menandai sumber daya Step Functions.

> **Note**
>
> Pembatasan penandaan tidak dapat ditingkatkan seperti kuota lainnya.

Pembatasan	Deskripsi
Jumlah maksimum tanda per sumber daya	50
Panjang kueri maksimum	128 karakter Unicode dalam UTF-8
Panjang nilai maksimum	256 karakter Unicode dalam UTF-8
Pembatasan prefiks	Jangan gunakan `aws:` awalan dalam nama atau nilai tag Anda karena itu dicadangkan untuk AWS digunakan. Anda tidak dapat mengedit atau menghapus nama atau nilai

Pembatasan	Deskripsi
	tanda dengan prefiks ini. Tanda dengan prefiks ini tidak memengaruhi tanda Anda per kuota sumber daya.
Pembatasan karakter	Tanda hanya boleh berisi huruf Unicode, angka, spasi putih, atau simbol berikut: `_ . : / = + - @`.

Pencatatan dan pemantauan di AWS Step Functions

Pencatatan dan pemantauan adalah bagian penting dari pemeliharaan keandalan, ketersediaan, dan performa Step Functions dan solusi AWS Anda. Ada beberapa alat yang tersedia untuk digunakan dengan Step Functions:

> **Tip**
>
> Untuk menerapkan alur kerja sampel ke Anda Akun AWS dan mempelajari cara memantau metrik, log, dan jejak eksekusi alur kerja, lihat Modul 12 - Observabilitas Lokakarya. AWS Step Functions

Topik

- Memantau Step Functions Menggunakan CloudWatch
- EventBridge (CloudWatch Events) untuk perubahan status eksekusi Step Functions
- Mencatat Step Functions Menggunakan AWS CloudTrail
- Logging menggunakanCloudWatchLog
- AWS X-Ray dan Step Functions
- Menggunakan Notifikasi Pengguna AWS dengan AWS Step Functions

Memantau Step Functions Menggunakan CloudWatch

Pemantauan adalah bagian penting dari menjaga keandalan, ketersediaan, dan kinerja AWS Step Functions dan AWS solusi Anda. Anda harus mengumpulkan sebanyak mungkin data pemantauan dari AWS layanan yang Anda gunakan sehingga Anda dapat men-debug kegagalan multi-titik. Namun sebelum mulai memantau Step Functions, Anda harus membuat rencana pemantauan yang mencakup jawaban atas pertanyaan berikut:

- Apa saja sasaran pemantauan Anda?
- Sumber daya apa yang akan Anda pantau?
- Seberapa sering Anda akan memantau sumber daya ini?
- Alat pemantauan apa yang akan Anda gunakan?
- Siapa yang akan melakukan tugas pemantauan?

- Siapa yang harus diberi tahu saat terjadi kesalahan?

Langkah selanjutnya adalah menetapkan dasar untuk kinerja normal di lingkungan Anda. Untuk melakukannya, ukur performa di berbagai waktu dan dalam syarat beban yang berbeda. Saat Anda memantau Step Functions, Anda harus mempertimbangkan untuk menyimpan data pemantauan historis. Data tersebut memberikan dasar untuk membandingkan data performa saat ini, mengidentifikasi pola performa normal dan anomali performa, dan merancang cara untuk mengatasi masalah.

Misalnya, dengan Step Functions, Anda dapat memantau berapa banyak aktivitas atau AWS Lambda tugas yang gagal karena batas waktu detak jantung. Ketika performa berada di luar garis dasar yang telah ditetapkan, Anda mungkin harus mengubah interval waktu Anda.

Untuk menetapkan acuan dasar, Anda harus, setidaknya, memantau metrik berikut:

- `ActivitiesStarted`
- `ActivitiesTimedOut`
- `ExecutionsStarted`
- `ExecutionsTimedOut`
- `LambdaFunctionsStarted`
- `LambdaFunctionsTimedOut`

Bagian berikut menjelaskan metrik yang disediakan Step Functions ke Amazon CloudWatch. Anda dapat menggunakan metrik ini untuk melacak aktivitas dan mesin status serta mengatur alarm pada nilai ambang batas. Anda dapat melihat metrik menggunakan. AWS Management Console

Metrik yang melaporkan interval waktu

Beberapa CloudWatch metrik Step Functions adalah interval waktu, selalu diukur dalam milidetik. Metrik ini umumnya sesuai dengan tahapan eksekusi Anda yang dapat diatur mesin status, aktivitas, dan timeout fungsi Lambda, dengan nama deskriptif.

Misalnya, metrik `ActivityRunTime` mengukur waktu yang dibutuhkan untuk suatu aktivitas selesai setelah mulai mengeksekusi. Anda dapat menetapkan nilai timeout untuk periode waktu yang sama.

Di CloudWatch konsol, Anda bisa mendapatkan hasil terbaik jika Anda memilih rata-rata sebagai statistik tampilan untuk metrik interval waktu.

Metrik yang melaporkan hitungan

Beberapa CloudWatch metrik Step Functions melaporkan hasil sebagai hitungan. Misalnya, `ExecutionsFailed` mencatat jumlah eksekusi mesin status yang gagal.

Step Functions memancarkan dua `ExecutionsStarted` metrik untuk setiap eksekusi mesin status. Hal ini menyebabkan SampleCountstatistik `ExecutionsStarted` metrik menunjukkan nilai 2 untuk setiap eksekusi mesin status. SampleCount Statistik menunjukkan `ExecutionStarted=1` dan `ExecutionStarted=0` kapan eksekusi selesai.

> **Tip**
>
> Sebaiknya pilih Jumlah sebagai statistik tampilan untuk metrik yang melaporkan hitungan di CloudWatch konsol.

Metrik eksekusi

`AWS/States`Namespace menyertakan metrik berikut untuk semua eksekusi Step Functions. Ini adalah metrik tanpa dimensi yang berlaku di seluruh akun Anda di suatu wilayah.

Metrik	Deskripsi
`OpenExecutionCount`	Perkiraan jumlah eksekusi yang saat ini terbuka —alur kerja yang sedang berlangsung di akun Anda. Tujuannya adalah untuk memberikan wawasan tentang kapan alur kerja Anda mendekati batas eksekusi maksimum, untuk menghindari ExecutionLimitExceededkesalahan saat memanggil `StartExecution` atau `RedriveExecution` untuk Alur Kerja Standar. Metrik tergantung pada transisi status alur kerja aktif, jadi pada tingkat rendah, perkiraan mungkin tidak selaras dengan jumlah alur kerja berjalan yang diamati.
`OpenExecutionLimit`	Jumlah maksimum eksekusi terbuka. Untuk informasi selengkapnya, lihat Kuota yang terkait dengan akun.

Metrik	Deskripsi
	Batas ini tidak berlaku untuk Alur Kerja Ekspres.

Metrik eksekusi untuk mesin status dengan versi atau alias

Ketika Anda menjalankan eksekusi mesin status dengan versi atau alias, Step Functions memancarkan metrik berikut. `ExecutionThrottled`Metrik hanya akan dipancarkan dalam kasus eksekusi terbatas. Metrik ini akan mencakup a `StateMachineArn` untuk mengidentifikasi mesin status tertentu.

Metrik	Deskripsi
`ExecutionTime`	Interval, dalam milidetik, antara waktu eksekusi dimulai dan waktu ditutup.
`ExecutionThrottled`	Jumlah `StateEntered` acara dan percobaan ulang yang telah dibatasi. Hal ini terkait dengan throttling `StateTransition` . Untuk informasi selengkapnya, lihat Kuota terkait throttling status.
`ExecutionsAborted`	Jumlah eksekusi yang dibatalkan atau dihentikan.
`ExecutionsFailed`	Jumlah eksekusi yang gagal.
`ExecutionsStarted`	Jumlah eksekusi yang dimulai.
`ExecutionsSucceeded`	Jumlah eksekusi yang berhasil diselesaikan.
`ExecutionsTimedOut`	Jumlah eksekusi yang habis waktu karena alasan apa pun.

Metrik eksekusi untuk Alur Kerja Ekspres

`AWS/States`Namespace menyertakan metrik berikut untuk eksekusi Step Functions Express Workflows'.

Metrik	Deskripsi
`ExpressExecutionMemory`	Total memori yang dikonsumsi oleh Alur Kerja Ekspres.
`ExpressExecutionBilledDuration`	Durasi di mana Alur Kerja Ekspres dibebankan.
`ExpressExecutionBilledMemory`	Jumlah memori yang dikonsumsi yang mengisi Alur Kerja Ekspres.

Redrivemetrik eksekusi untuk Alur Kerja Standar

Saat Anda redrivemelakukan eksekusi mesin status, Step Functions memancarkan metrik berikut.

Untuk semua redriven eksekusi, `Executions*` metrik dipancarkan. Misalnya, katakanlah redriven eksekusi dibatalkan. Eksekusi ini akan memancarkan titik data bukan nol untuk keduanya dan. `RedrivenExecutionsAborted` `ExecutionsAborted`

Metrik	Deskripsi
`ExecutionsRedriven`	Jumlah redriven eksekusi.
`RedrivenExecutionsAborted`	Jumlah redriven eksekusi yang dibatalkan atau dihentikan.
`RedrivenExecutionsTimedOut`	Jumlah redriven eksekusi yang habis waktu karena alasan apa pun.
`RedrivenExecutionsSucceeded`	Jumlah redriven eksekusi yang berhasil diselesaikan.
`RedrivenExecutionsFailed`	Jumlah redriven eksekusi yang gagal.

Dimensi untuk metrik eksekusi Step Functions

Dimensi	Deskripsi
`StateMachineArn`	Amazon Resource Name (ARN) mesin status untuk eksekusi yang dimaksud.

Dimensi untuk eksekusi dengan versi

Dimensi	Deskripsi
`StateMachineArn`	Nama Sumber Daya Amazon (ARN) dari mesin negara yang eksekusinya dimulai oleh versi.
`Version`	Versi mesin negara digunakan untuk memulai eksekusi.

Dimensi untuk eksekusi dengan alias

Dimensi	Deskripsi
`StateMachineArn`	Nama Sumber Daya Amazon (ARN) dari mesin negara yang eksekusinya dimulai dengan alias.
`Alias`	Alias mesin negara digunakan untuk memulai eksekusi.

Metrik jumlah sumber daya untuk versi dan alias

`AWS/States`Namespace menyertakan metrik berikut untuk jumlah versi dan alias mesin status.

Metrik	Deskripsi
AliasCount	Jumlah alias yang dibuat untuk mesin negara. Anda dapat membuat hingga 100 alias untuk setiap mesin negara.

Metrik	Deskripsi
VersionCount	Jumlah versi yang diterbitkan untuk mesin negara. Anda dapat mempublikasikan hingga 1000 versi mesin negara.

Dimensi untuk metrik jumlah sumber daya untuk versi dan alias

Dimensi	Deskripsi
`ResourceArn`	Nama Sumber Daya Amazon (ARN) dari mesin negara bagian dengan versi atau alias.

Metrik Aktivitas

Namespace `AWS/States` mencakup metrik berikut untuk aktivitas Step Functions.

Metrik	Deskripsi
`ActivityRunTime`	Interval, dalam milidetik, antara waktu aktivitas dimulai dan waktu ditutup.
`ActivityScheduleTime`	Interval, dalam milidetik, di mana aktivitas tetap dalam status jadwal.
`ActivityTime`	Interval, dalam milidetik, antara waktu aktivitas dijadwalkan dan waktu ditutup.
`ActivitiesFailed`	Jumlah kegiatan yang gagal.
`ActivitiesHeartbeatTimedOut`	Jumlah aktivitas yang time out karena batas waktu detak jantung.
`ActivitiesScheduled`	Jumlah kegiatan yang dijadwalkan.
`ActivitiesStarted`	Jumlah kegiatan yang dimulai.

Metrik	Deskripsi
`ActivitiesSucceeded`	Jumlah kegiatan yang berhasil diselesaikan.
`ActivitiesTimedOut`	Jumlah kegiatan yang time out pada waktu dekat.

Dimensi untuk Metrik Aktivitas Step Functions

Dimensi	Deskripsi
`ActivityArn`	ARN aktivitas.

Metrik Fungsi Lambda

Namespace `AWS/States` mencakup metrik berikut untuk fungsi Lambda Step Functions.

Metrik	Deskripsi
`LambdaFunctionRunTime`	Interval, dalam milidetik, antara waktu fungsi Lambda dimulai dan waktu ditutup.
`LambdaFunctionScheduleTime`	Interval, dalam milidetik, di mana fungsi Lambda tetap dalam status jadwal.
`LambdaFunctionTime`	Interval, dalam milidetik, antara waktu fungsi Lambda dijadwalkan dan waktu ditutup.
`LambdaFunctionsFailed`	Jumlah fungsi Lambda yang gagal.
`LambdaFunctionsScheduled`	Jumlah fungsi Lambda terjadwal.
`LambdaFunctionsStarted`	Jumlah fungsi Lambda yang dimulai.

Metrik	Deskripsi
`LambdaFunctionsSucceeded`	Jumlah fungsi Lambda yang berhasil diselesaikan.
`LambdaFunctionsTimedOut`	Jumlah fungsi Lambda yang habis waktu tutup.

Dimensi untuk Metrik Fungsi Lambda Step Functions

Dimensi	Deskripsi
`LambdaFunctionArn`	ARN fungsi Lambda.

> **Note**
>
> Metrik Fungsi Lambda dipancarkan untuk status Tugas yang menentukan ARN fungsi Lambda di kolom `Resource`. Status tugas yang menggunakan `"Resource": "arn:aws:states:::lambda:invoke"` memancarkan Metrik Integrasi Layanan sebagai gantinya. Untuk informasi selengkapnya, lihat Panggil Lambda dengan Step Functions.

Metrik Integrasi Layanan

Namespace `AWS/States` mencakup metrik berikut untuk integrasi layanan Step Functions. Untuk informasi selengkapnya, lihat Menggunakan AWS Step Functions dengan layanan lain.

Metrik	Deskripsi
`ServiceIntegrationRunTime`	Interval, dalam milidetik, antara waktu Tugas Layanan dimulai dan waktu ditutup.
`ServiceIntegrationScheduleTime`	Interval, dalam milidetik, di mana Tugas Layanan tetap dalam status jadwal.

Metrik	Deskripsi
`ServiceIntegrationTime`	Interval, dalam milidetik, antara waktu Tugas Layanan dijadwalkan dan waktu ditutup.
`ServiceIntegrationsFailed`	Jumlah Tugas Layanan yang gagal.
`ServiceIntegrationsScheduled`	Jumlah Tugas Layanan yang dijadwalkan.
`ServiceIntegrationsStarted`	Jumlah Tugas Layanan yang dimulai.
`ServiceIntegrationsSucceeded`	Jumlah Tugas Layanan yang berhasil diselesaikan.
`ServiceIntegrationsTimedOut`	Jumlah Tugas Layanan yang habis waktu tutup.

Dimensi untuk Metrik Integrasi Layanan Step Functions

Dimensi	Deskripsi
`ServiceIntegrationResourceArn`	Sumber daya ARN dari layanan terintegrasi.

Metrik Layanan

Namespace `AWS/States` mencakup metrik berikut untuk layanan Step Functions.

Metrik	Deskripsi
`ThrottledEvents`	Hitungan permintaan yang telah dibatasi.
`ProvisionedBucketSize`	Hitungan permintaan yang tersedia per detik.

Metrik	Deskripsi
`ProvisionedRefillRate`	Hitungan permintaan per detik yang diizinkan masuk ke dalam ember.
`ConsumedCapacity`	Hitungan permintaan per detik.

Dimensi untuk Metrik Layanan Step Functions

Dimensi	Deskripsi
`ServiceMetric`	Memfilter data untuk menampilkan metrik Transisi Status.

Metrik API

Namespace `AWS/States` mencakup metrik berikut untuk API Step Functions.

Metrik	Deskripsi
`ThrottledEvents`	Hitungan permintaan yang telah dibatasi.
`ProvisionedBucketSize`	Hitungan permintaan yang tersedia per detik.
`ProvisionedRefillRate`	Hitungan permintaan per detik yang diizinkan masuk ke dalam ember.
`ConsumedCapacity`	Hitungan permintaan per detik.

Dimensi untuk Metrik API Step Functions

Dimensi	Deskripsi
`APIName`	Memfilter data ke API dari nama API yang ditentukan.

Pengiriman CloudWatch metrik upaya terbaik

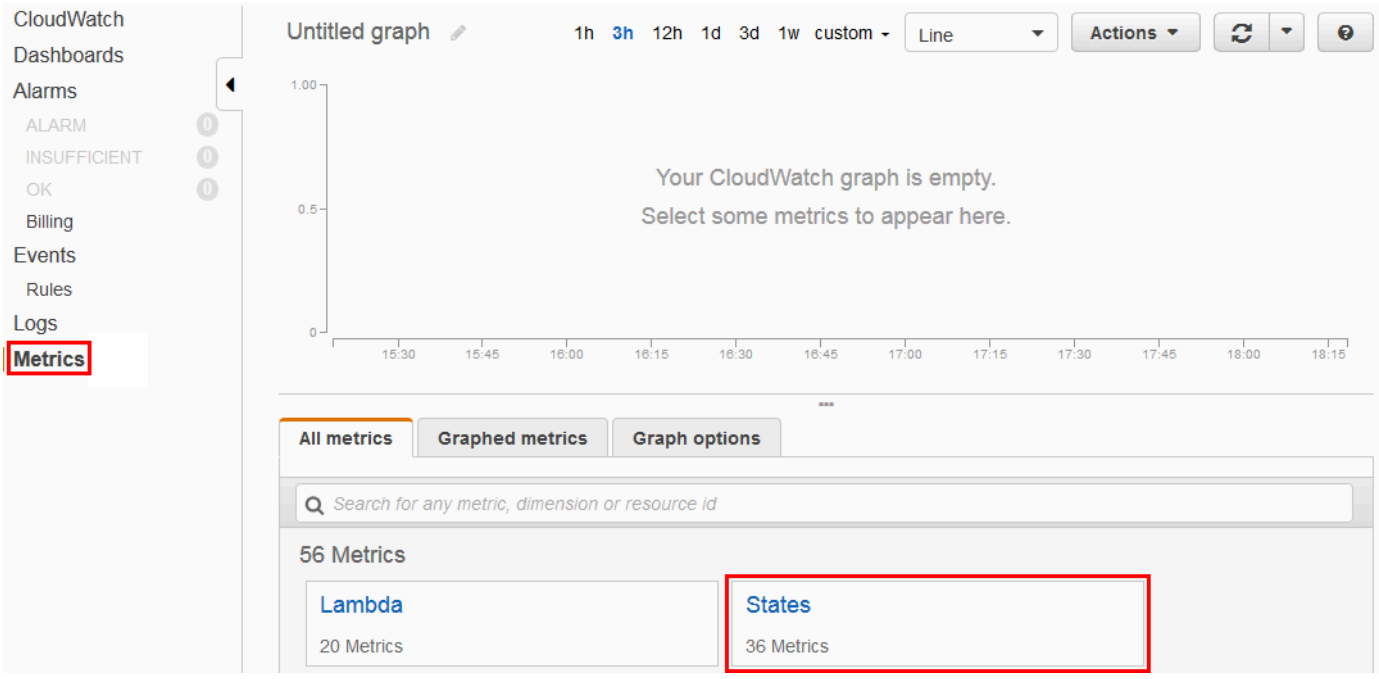
Metrik CloudWatch diberikan dengan dasar upaya terbaik.

Kelengkapan dan ketepatan waktu metrik tidak dijamin. Titik data untuk permintaan tertentu mungkin dikembalikan dengan stempel waktu yang lebih lambat daripada ketika permintaan tersebut sebenarnya diproses. Titik data selama satu menit mungkin tertunda sebelum tersedia CloudWatch, atau mungkin tidak dikirimkan sama sekali. CloudWatchmetrik permintaan memberi Anda gambaran tentang eksekusi mesin status dalam waktu hampir nyata. Ini tidak dimaksudkan untuk menjadi akuntansi lengkap dari semua metrik terkait eksekusi.

Ini mengikuti sifat upaya terbaik dari fitur ini bahwa laporan yang tersedia di Dasbor Manajemen Penagihan & Biaya mungkin mencakup satu atau lebih permintaan akses yang tidak muncul dalam metrik eksekusi.

Melihat Metrik untuk Step Functions

1. Masuk ke AWS Management Console dan buka CloudWatch konsol.
2. Pilih Metrik, dan pada tab Semua Metrik, pilih Status.

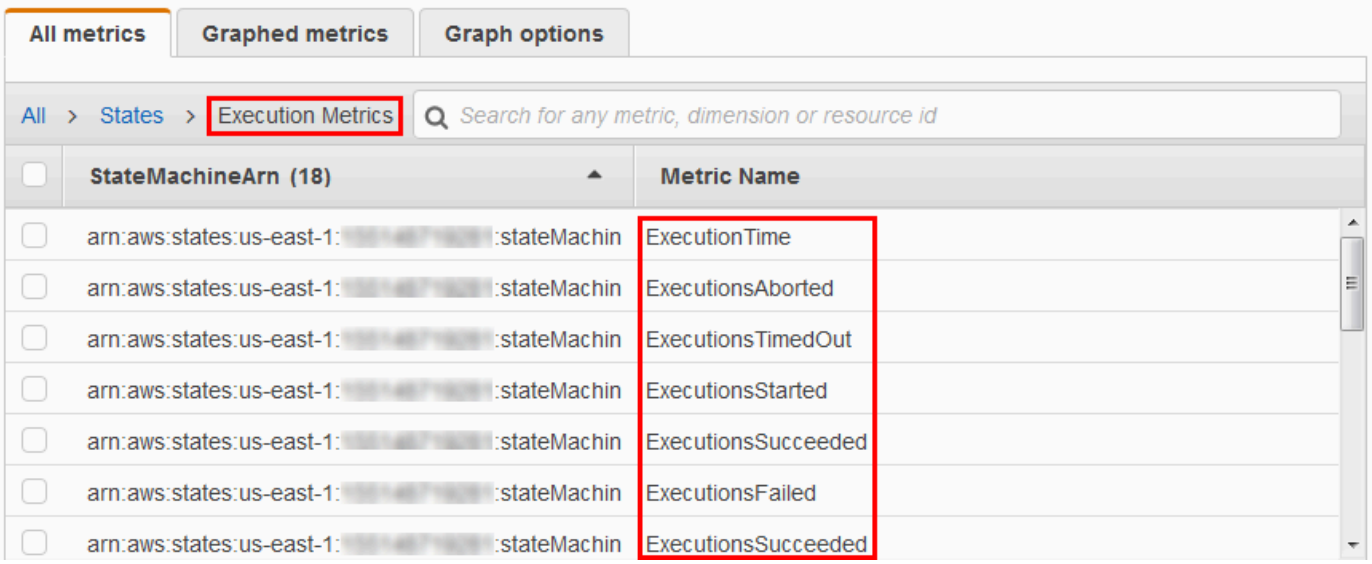


Jika Anda menjalankan eksekusi baru-baru ini, Anda akan melihat hingga empat tipe metrik:

- Metrik Eksekusi
- Metrik Fungsi Aktivitas

- Metrik Fungsi Lambda
- Metrik Integrasi Layanan

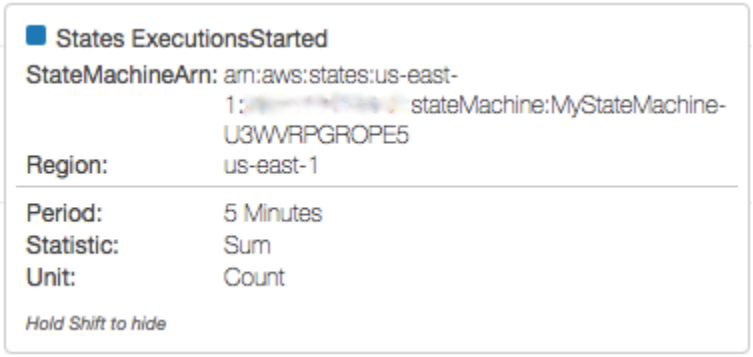
3. Pilih tipe metrik untuk melihat daftar metrik.



- Untuk mengurutkan metrik Anda berdasarkan Nama Metrik atau StateMachineArn, gunakan judul kolom.
- Untuk melihat grafik metrik, pilih kotak di samping metrik pada daftar. Anda dapat mengubah parameter grafik menggunakan kontrol rentang waktu di atas tampilan grafik.

  Anda dapat memilih rentang waktu kustom menggunakan nilai relatif atau absolut (hari dan waktu tertentu). Anda juga dapat menggunakan daftar menurun untuk menampilkan nilai sebagai garis, daerah yang ditumpuk, atau nomor (nilai).
- Untuk melihat detail tentang grafik, arahkan kursor ke kode warna metrik yang muncul di bawah grafik.

  Detail metrik akan ditampilkan.



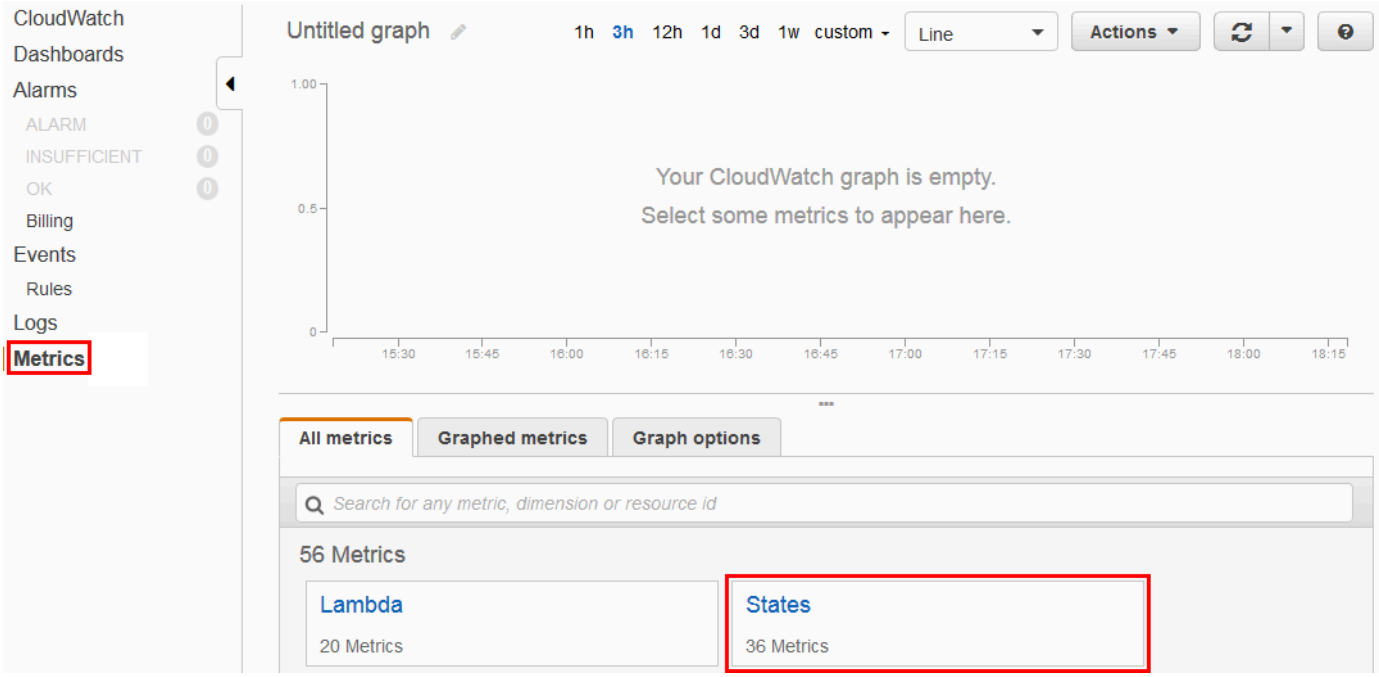
Untuk informasi selengkapnya tentang bekerja dengan CloudWatch metrik, lihat Menggunakan CloudWatch Metrik Amazon di CloudWatch Panduan Pengguna Amazon.

Pengaturan Alarm untuk Step Functions

Anda dapat menggunakan CloudWatch alarm Amazon untuk melakukan tindakan. Misalnya, jika Anda ingin tahu saat ambang alarm tercapai, Anda dapat mengatur alarm untuk mengirim notifikasi ke topik Amazon SNS atau mengirim email ketika metrik `StateMachinesFailed` naik di atas ambang tertentu.

Untuk mengatur alarm pada metrik

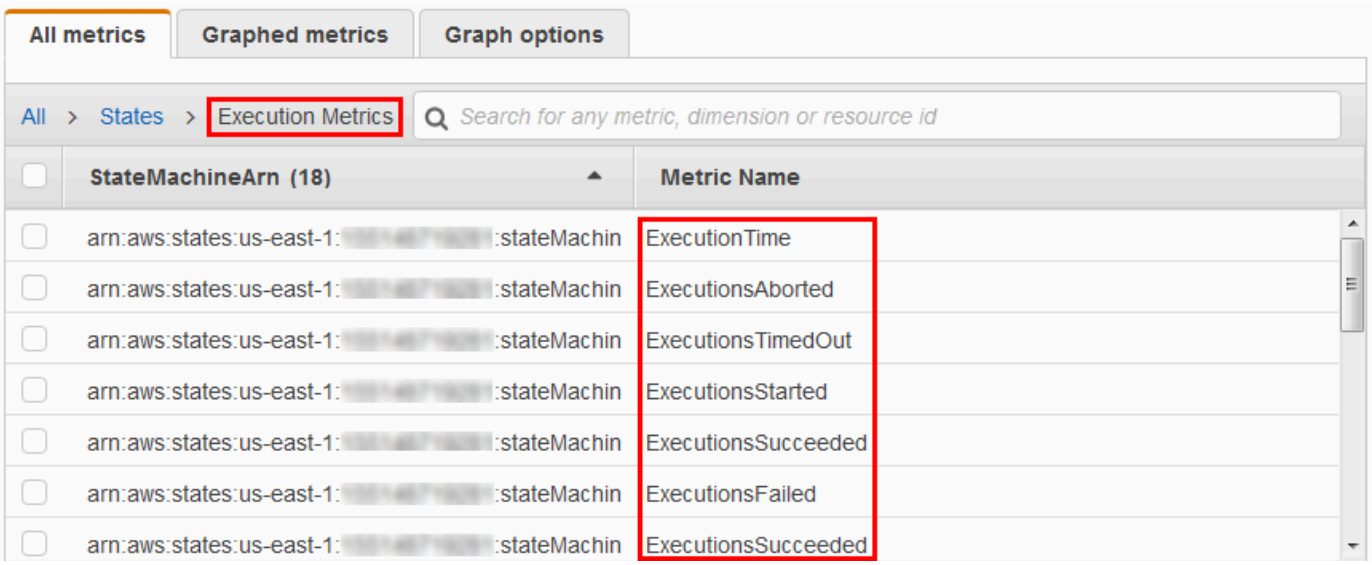
1. Masuk ke AWS Management Console dan buka CloudWatch konsol.
2. Pilih Metrik, dan pada tab Semua Metrik, pilih Status.



Jika Anda menjalankan eksekusi baru-baru ini, Anda akan melihat hingga empat tipe metrik:

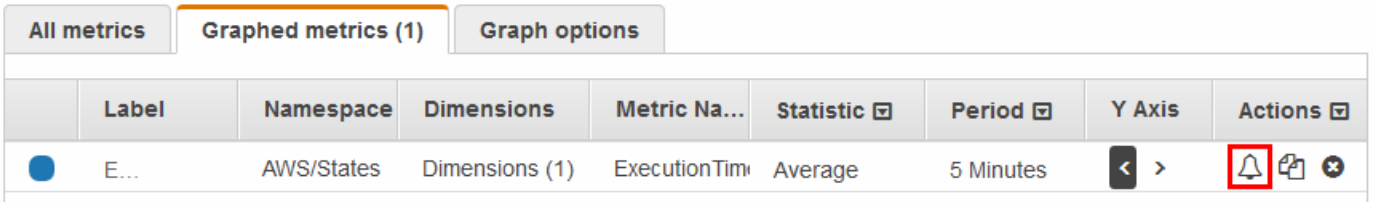
- Metrik Eksekusi
- Metrik Fungsi Aktivitas
- Metrik Fungsi Lambda
- Metrik Integrasi Layanan

3. Pilih tipe metrik untuk melihat daftar metrik.

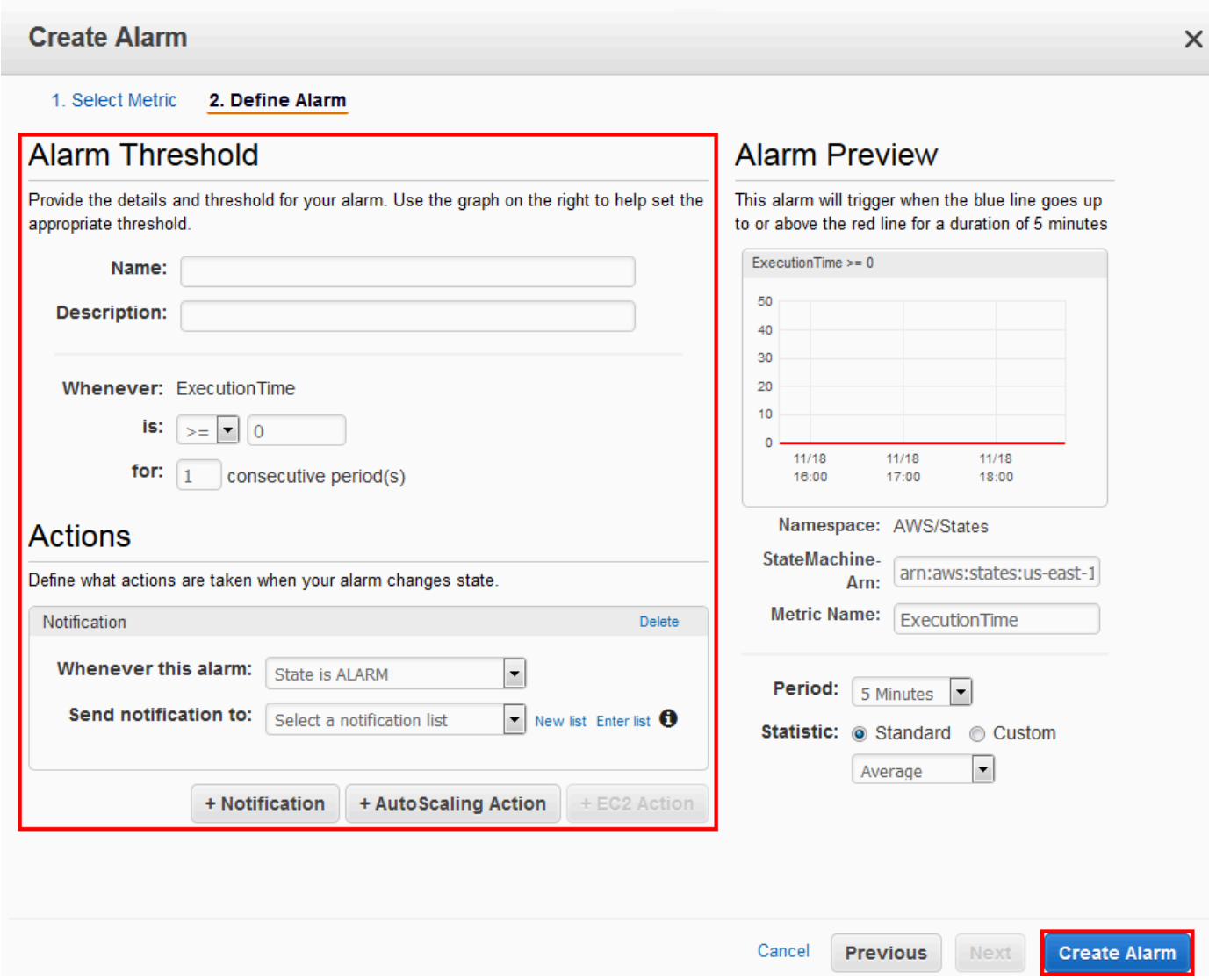


4. Pilih metrik, lalu pilih Metrik Digrafikkan.
5. Pilih

di samping metrik pada daftar.



Halaman Buat alarm ditampilkan.



6. Masukkan nilai untuk Ambang batas alarm dan Tindakan, lalu pilih Buat Alarm.

Untuk informasi selengkapnya tentang pengaturan dan penggunaan CloudWatch alarm, lihat Membuat CloudWatch Alarm Amazon di CloudWatch Panduan Pengguna Amazon.

EventBridge (CloudWatch Events) untuk perubahan status eksekusi Step Functions

Amazon EventBridge adalah AWS layanan yang memungkinkan Anda merespons perubahan status dalam AWS sumber daya. Misalnya, Anda dapat menanggapi perubahan status eksekusi dari Alur Kerja Standar Step Functions dengan EventBridge menggunakan dua cara berikut:

- Anda dapat mengonfigurasi EventBridge aturan untuk bereaksi terhadap peristiwa yang dipancarkan saat status eksekusi mesin status Step Functions berubah. Hal ini memungkinkan Anda untuk memantau alur kerja Anda tanpa harus terus-menerus melakukan polling menggunakan API DescribeExecution. Berdasarkan perubahan dalam eksekusi mesin status, Anda dapat menggunakan EventBridge target untuk memulai eksekusi mesin status baru, AWS Lambda fungsi panggilan, mempublikasikan pesan ke topik Amazon Simple Notification Service (Amazon SNS), dan banyak lagi.
- Anda juga dapat mengonfigurasi mesin status Step Functions sebagai target EventBridge. Hal ini mengaktifkan Anda untuk memicu eksekusi dari alur kerja Step Functions dalam menanggapi sebuah peristiwa dari layanan AWS yang lain.

  Untuk informasi selengkapnya, lihat Panduan EventBridge Pengguna Amazon.

Alur Kerja Ekspres, bagaimanapun, tidak memancarkan peristiwa ke. EventBridge Untuk memantau eksekusi Alur Kerja Ekspres, Anda dapat menggunakan CloudWatch Log. Untuk melakukan ini, pada halaman Detail Eksekusi mesin negara, pilih tab Pemantauan dan Pencatatan. Pada tab Monitoring, Anda dapat melihat CloudWatch metrik untuk peristiwa, seperti Durasi Eksekusi, Kesalahan Eksekusi, dan Memori Tagihan. Pada tab Logging, Anda dapat melihat log terbaru dan konfigurasi logging.

> **Tip**
>
> Untuk menerapkan contoh Alur Kerja Ekspres ke Anda Akun AWS dan mempelajari cara memantau Alur Kerja Ekspres, lihat modul Pemantauan Alur Kerja Ekspres dari Lokakarya. AWS Step Functions

EventBridge muatan

Sebuah EventBridge peristiwa dapat berisi properti input dalam definisinya. Untuk beberapa peristiwa, suatu EventBridge peristiwa juga dapat berisi properti output dalam definisinya.

- Jika input lolos gabungan dan output keluar yang dikirim EventBridge melebihi 248KB, maka input akan dikecualikan. Demikian pula, jika output escape melebihi 248KB, output akan dikecualikan. Ini adalah hasil dari kuota EventBridge acara.

- Anda dapat menentukan apakah muatan telah dipotong dengan properti `inputDetails` dan `outputDetails`. Untuk informasi selengkapnya, lihat Tipe Data `CloudWatchEventsExecutionDataDetails`.
- Untuk Alur Kerja Standar, Anda dapat melihat input dan output lengkap dengan menggunakan DescribeExecution.
- `DescribeExecution` ini tidak tersedia untuk Alur Kerja Ekspres. Jika ingin melihat input/output penuh, Anda dapat membungkus Alur Kerja Ekspres dengan Alur Kerja Standar. Pilihan lainnya adalah menggunakan ARN Amazon S3. Untuk informasi selengkapnya tentang menggunakan ARN, lihat the section called “Gunakan ARN Amazon S3 bukan meneruskan muatan besar”.

Topik

- Contoh peristiwa Step Functions
- Merutekan acara Step Functions ke EventBridge konsol EventBridge

Contoh peristiwa Step Functions

Berikut ini adalah contoh Step Functions mengirim peristiwa ke EventBridge:

Topik

- Eksekusi dimulai
- Eksekusi berhasil
- Eksekusi gagal
- Waktu eksekusi habis
- Eksekusi dibatalkan

Dalam setiap kasus, bagian `detail` dalam data peristiwa menyediakan informasi yang sama sebagai API `DescribeExecution`. Bidang `status` menunjukkan status eksekusi pada saat peristiwa dikirim, salah satu `RUNNING`, `SUCCEEDED`, `FAILED`, `TIMED_OUT`, atau `ABORTED` tergantung pada peristiwa yang dipancarkan.

Eksekusi dimulai

```
{
    "version": "0",
    "id": "315c1398-40ff-a850-213b-158f73e60175",
```

```
    "detail-type": "Step Functions Execution Status Change",
    "source": "aws.states",
    "account": "123456789012",
    "time": "2019-02-26T19:42:21Z",
    "region": "us-east-2",
    "resources": [
      "arn:aws:states:us-east-2:123456789012:execution:state-machine-name:execution-
name"
    ],
    "detail": {
        "executionArn": "arn:aws:states:us-east-2:123456789012:execution:state-machine-
name:execution-name",
        "stateMachineArn": "arn:aws::states:us-east-2:123456789012:stateMachine:state-
machine",
        "name": "execution-name",
        "status": "RUNNING",
        "startDate": 1551225271984,
        "stopDate":  null,
        "input": "{}",
        "inputDetails": {
             "included": true
        },
        "output": null,
        "outputDetails": null
    }
}
```

Eksekusi berhasil

```
{
    "version": "0",
    "id": "315c1398-40ff-a850-213b-158f73e60175",
    "detail-type": "Step Functions Execution Status Change",
    "source": "aws.states",
    "account": "123456789012",
    "time": "2019-02-26T19:42:21Z",
    "region": "us-east-2",
    "resources": [
      "arn:aws:states:us-east-2:123456789012:execution:state-machine-name:execution-
name"
    ],
    "detail": {
```

```
        "executionArn": "arn:aws:states:us-east-2:123456789012:execution:state-machine-
name:execution-name",
        "stateMachineArn": "arn:aws:states:us-east-2:123456789012:stateMachine:state-
machine",
        "name": "execution-name",
        "status": "SUCCEEDED",
        "startDate": 1547148840101,
        "stopDate": 1547148840122,
        "input": "{}",
        "inputDetails": {
             "included": true
        },
        "output": "\"Hello World!\"",
        "outputDetails": {
             "included": true
        }
    }
}
```

Eksekusi gagal

```
{
    "version": "0",
    "id": "315c1398-40ff-a850-213b-158f73e60175",
    "detail-type": "Step Functions Execution Status Change",
    "source": "aws.states",
    "account": "123456789012",
    "time": "2019-02-26T19:42:21Z",
    "region": "us-east-2",
    "resources": [
      "arn:aws:states:us-east-2:123456789012:execution:state-machine-name:execution-
name"
    ],
    "detail": {
        "executionArn": "arn:aws:states:us-east-2:123456789012:execution:state-machine-
name:execution-name",
        "stateMachineArn": "arn:aws:states:us-east-2:123456789012:stateMachine:state-
machine",
        "name": "execution-name",
        "status": "FAILED",
        "startDate": 1551225146847,
        "stopDate": 1551225151881,
        "input": "{}",
```

```
        "inputDetails": {
             "included": true
        },
        "output": null,
        "outputDetails": null
    }
}
```

Waktu eksekusi habis

```
{
    "version": "0",
    "id": "315c1398-40ff-a850-213b-158f73e60175",
    "detail-type": "Step Functions Execution Status Change",
    "source": "aws.states",
    "account": "123456789012",
    "time": "2019-02-26T19:42:21Z",
    "region": "us-east-2",
    "resources": [
      "arn:aws:states:us-east-2:123456789012:execution:state-machine-name:execution-
name"
    ],
    "detail": {
        "executionArn": "arn:aws:states:us-east-2:123456789012:execution:state-machine-
name:execution-name",
        "stateMachineArn": "arn:aws:states:us-east-2:123456789012:stateMachine:state-
machine",
        "name": "execution-name",
        "status": "TIMED_OUT",
        "startDate": 1551224926156,
        "stopDate": 1551224927157,
        "input": "{}",
        "inputDetails": {
             "included": true
        },
        "output": null,
        "outputDetails": null
    }
}
```

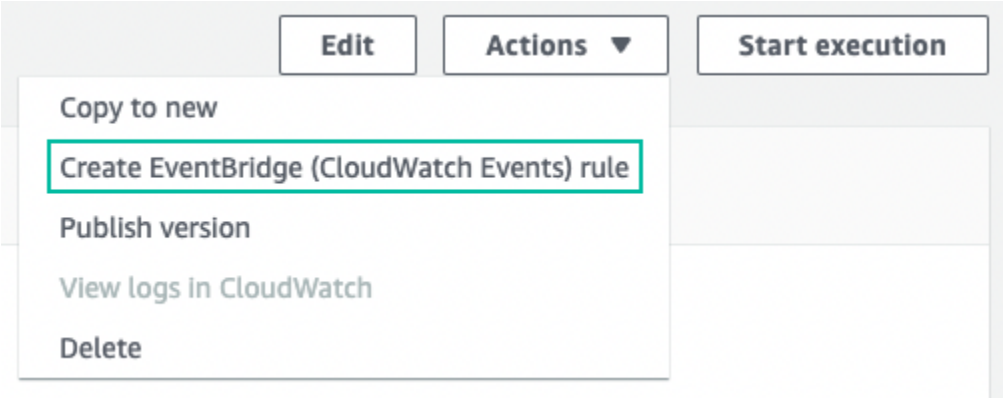
Eksekusi dibatalkan

```
{
    "version": "0",
    "id": "315c1398-40ff-a850-213b-158f73e60175",
    "detail-type": "Step Functions Execution Status Change",
    "source": "aws.states",
    "account": "123456789012",
    "time": "2019-02-26T19:42:21Z",
    "region": "us-east-2",
    "resources": [
      "arn:aws:states:us-east-2:123456789012:execution:state-machine-name:execution-
name"
    ],
    "detail": {
        "executionArn": "arn:aws:states:us-east-2:123456789012:execution:state-machine-
name:execution-name",
        "stateMachineArn": "arn:aws:states:us-east-2:123456789012:stateMachine:state-
machine",
        "name": "execution-name",
        "status": "ABORTED",
        "startDate": 1551225014968,
        "stopDate": 1551225017576,
        "input": "{}",
        "inputDetails": {
             "included": true
        },
        "output": null,
        "outputDetails": null
    }
}
```

Merutekan acara Step Functions ke EventBridge konsol EventBridge

Gunakan petunjuk berikut untuk mempelajari cara memicu eksekusi mesin status Step Functions setiap kali mesin status Step Functions tertentu selesai berjalan dengan sukses. Anda menggunakan EventBridge konsol Amazon untuk menentukan mesin status yang eksekusinya ingin Anda picu.

1. Pada halaman Detail mesin status, pilih Tindakan, lalu pilih aturan Buat EventBridge (CloudWatch Peristiwa).



Atau, buka EventBridge konsol di https://console.aws.amazon.com/events/. Di panel navigasi, pilih Aturan di bawah Bus.

2. Pilih Buat aturan. Ini membuka halaman Define rule detail.
3. Masukkan Nama untuk aturan Anda (misalnya,*StepFunctionsEventRule*) dan secara opsional masukkan Deskripsi untuk aturan tersebut.
4. Untuk bus Acara dan tipe Aturan, pertahankan pilihan default.
5. Pilih Berikutnya. Ini membuka halaman pola acara Build.
6. Di bawah Sumber Acara, pertahankan pilihan default AWS acara atau acara EventBridge mitra.
7. Simpan pilihan default untuk acara Sampel dan bagian metode Pembuatan.
8. Di bawah pola Acara, lakukan hal berikut:
   a. Dalam daftar dropdown sumber acara, pertahankan pilihan layanan default. AWS
   b. Dari daftar dropdown AWS layanan, pilih Step Functions.
   c. Dari daftar dropdown tipe Event, pilih Step Functions Execution Status Change.
   d. (Opsional) Konfigurasikan status tertentu, status mesin Amazon Resource Name (ARN), atau eksekusi ARN. Untuk prosedur ini, pilih Status spesifik, lalu pilih BERHASIL dari daftar dropdown.
9. Pilih Berikutnya. Ini membuka halaman Select target (s).
10. Di bawah Jenis target, pertahankan pilihan AWS layanan default.
11. Dari daftar tarik-turun Pilih target, pilih layanan. AWS Misalnya, Anda dapat meluncurkan fungsi Lambda, atau menjalankan mesin status Step Functions. Untuk prosedur ini, pilih Step Functions state machine.
12. Dari daftar dropdown mesin Negara, pilih mesin negara.

13. Di bawah peran Eksekusi, pertahankan pilihan default Buat peran baru untuk sumber daya khusus ini.
14. Pilih Berikutnya. Ini membuka halaman Konfigurasi tag.
15. pilih Next lagi. Ini membuka halaman Review dan create.
16. Tinjau detail aturan dan pilih Buat aturan.

    Aturan dibuat dan halaman Aturan ditampilkan, mencantumkan semua EventBridge aturan Amazon Anda.

Mencatat Step Functions Menggunakan AWS CloudTrail

Step Functions terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau layanan AWS di Step Functions. CloudTrailmenangkap semua panggilan API untuk Step Functions sebagai event, termasuk panggilan dari konsol Step Functions dan dari panggilan kode ke API Step Functions.

Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail peristiwa berkelanjutan ke bucket Amazon Simple Storage Service (Amazon S3), termasuk peristiwa untuk Step Functions. Jika Anda tidak membuat konfigurasi jejak, Anda masih dapat melihat kejadian terbaru dalam konsol CloudTrail di Riwayat peristiwa.

Menggunakan informasi yang dikumpulkan olehCloudTrail, Anda dapat menentukan permintaan yang dibuat untuk Step Functions, alamat IP dari mana permintaan dibuat, yang membuat permintaan, ketika dibuat, dan rincian tambahan.

Untuk mempelajari lebih lanjutCloudTrail, lihat Panduan AWS CloudTrail Pengguna.

Langkah Fungsi Informasi di CloudTrail

CloudTrail diaktifkan pada akun AWS Anda saat Anda membuat akun tersebut. Ketika aktivitas terjadi di Step Functions, aktivitas tersebut direkam dalam CloudTrail peristiwa dengan peristiwa AWS layanan lain dalam riwayat peristiwa.

Anda dapat melihat, mencari, dan mengunduh peristiwa terbaru di akun AWS Anda. Untuk informasi lain, lihat Melihat Peristiwa dengan Riwayat Peristiwa CloudTrail.

Untuk catatan berkelanjutan tentang peristiwa di akun AWS Anda, termasuk peristiwa untuk Step Functions, buat jejak. Jejak memungkinkan CloudTrail untuk mengirim file log ke bucket Amazon S3. Secara default, saat Anda membuat jejak di dalam konsol tersebut, jejak diterapkan ke semua

Wilayah AWS. Jejak mencatat peristiwa dari semua Wilayah di partisi AWS dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi AWS layanan lain untuk menganalisis lebih lanjut dan bertindak atas data peristiwa yang dikumpulkan dalam CloudTrail log.

Untuk informasi selengkapnya, lihat yang berikut:

- Ikhtisar untuk Membuat Jejak
- CloudTrailLayanan dan Integrasi yang Didukung
- Mengonfigurasi Pemberitahuan Amazon SNS untuk CloudTrail
- Menerima CloudTrail Berkas Log dari Beberapa Wilayah dan Menerima CloudTrail Berkas Log dari Beberapa Akun

Setiap entri peristiwa atau log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan berikut ini:

- Apakah permintaan dibuat dengan kredensi root atau pengguna
- Jika permintaan tersebut dibuat dengan kredensial keamanan sementara untuk peran atau pengguna gabungan
- Jika permintaan tersebut dibuat oleh layanan AWS lainnya

Untuk informasi lain, lihat Elemen userIdentity CloudTrail.

Contoh: Entri Berkas Log Step Functions

Jejak adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai berkas log ke bucket Amazon S3 yang telah Anda tentukan. Berkas log CloudTrail berisi satu atau beberapa entri log. Peristuwa mewakili satu permintaan dari sumber apa pun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. CloudTrailfile log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, sehingga tidak muncul dalam urutan tertentu.

CreateActivity

Contoh berikut menunjukkan catatan log CloudTrail yang menunjukkan tindakan `CreateActivity`.

```
{
    "eventVersion": "1.04",
```

```
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAJYDLDBVBI4EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/test-user",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "test-user"
    },
    "eventTime": "2016-10-28T01:17:56Z",
    "eventSource": "states.amazonaws.com",
    "eventName": "CreateActivity",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "name":
 "OtherActivityPrefix.2016-10-27-18-16-56.894c791e-2ced-4cf4-8523-376469410c25"
    },
    "responseElements": {
        "activityArn": "arn:aws:states:us-
east-1:123456789012:activity:OtherActivityPrefix.2016-10-27-18-16-56.894c791e-2ced-4cf4-8523-37
        "creationDate": "Oct 28, 2016 1:17:56 AM"
    },
    "requestID": "37c67602-9cac-11e6-aed5-5b57d226e9ef",
    "eventID": "dc3becef-d06d-49bf-bc93-9b76b5f00774",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

CreateStateMachine

Contoh berikut menunjukkan catatan log CloudTrail yang menunjukkan tindakan `CreateStateMachine`.

```
{
    "eventVersion": "1.04",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAJYDLDBVBI4EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/test-user",
        "accountId": "123456789012",
        "accessKeyId": "AKIAJL5C75K4ZEXAMPLE",
        "userName": "test-user"
```

```
    },
    "eventTime": "2016-10-28T01:18:07Z",
    "eventSource": "states.amazonaws.com",
    "eventName": "CreateStateMachine",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "name": "testUser.2016-10-27-18-17-06.bd144e18-0437-476e-9bb",
        "roleArn": "arn:aws:iam::123456789012:role/graphene/tests/graphene-execution-
role",
        "definition": "{   \"StartAt\": \"SinglePass\",   \"States\": {
  \"SinglePass\": {           \"Type\": \"Pass\",           \"End\": true       }   }}"
    },
    "responseElements": {
        "stateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:testUser.2016-10-27-18-17-06.bd144e18-0437-476e-9bb",
        "creationDate": "Oct 28, 2016 1:18:07 AM"
    },
    "requestID": "3da6370c-9cac-11e6-aed5-5b57d226e9ef",
    "eventID": "84a0441d-fa06-4691-a60a-aab9e46d689c",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

DeleteActivity

Contoh berikut menunjukkan catatan log CloudTrail yang menunjukkan tindakan `DeleteActivity`.

```
{
      "eventVersion": "1.04",
      "userIdentity": {
          "type": "IAMUser",
          "principalId": "AIDAJYDLDBVBI4EXAMPLE",
          "arn": "arn:aws:iam::123456789012:user/test-user",
          "accountId": "123456789012",
          "accessKeyId": "AKIAJL5C75K4ZEXAMPLE",
          "userName": "test-user"
      },
      "eventTime": "2016-10-28T01:18:27Z",
      "eventSource": "states.amazonaws.com",
      "eventName": "DeleteActivity",
      "awsRegion": "us-east-1",
```

```
      "sourceIPAddress": "AWS Internal",
      "userAgent": "AWS Internal",
      "requestParameters": {
          "activityArn": "arn:aws:states:us-
east-1:123456789012:activity:testUser.2016-10-27-18-11-35.f017c391-9363-481a-be2e-"
      },
      "responseElements": null,
      "requestID": "490374ea-9cac-11e6-aed5-5b57d226e9ef",
      "eventID": "e5eb9a3d-13bc-4fa1-9531-232d1914d263",
      "eventType": "AwsApiCall",
      "recipientAccountId": "123456789012"
  }
```

DeleteStateMachine

Contoh berikut menunjukkan catatan log CloudTrail yang menunjukkan tindakan `DeleteStateMachine`.

```
{
    "eventVersion": "1.04",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAJABK5MNKNAEXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/graphene/tests/test-user",
        "accountId": "123456789012",
        "accessKeyId": "AKIAJA2ELRVCPEXAMPLE",
        "userName": "test-user"
    },
    "eventTime": "2016-10-28T01:17:37Z",
    "eventSource": "states.amazonaws.com",
    "eventName": "DeleteStateMachine",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "errorCode": "AccessDenied",
    "errorMessage": "User: arn:aws:iam::123456789012:user/graphene/tests/test-user is
 not authorized to perform: states:DeleteStateMachine on resource: arn:aws:states:us-
east-1:123456789012:stateMachine:testUser.2016-10-27-18-16-38.ec6e261f-1323-4555-9fa",
    "requestParameters": null,
    "responseElements": null,
    "requestID": "2cf23f3c-9cac-11e6-aed5-5b57d226e9ef",
    "eventID": "4a622d5c-e9cf-4051-90f2-4cdb69792cd8",
    "eventType": "AwsApiCall",
```

```
    "recipientAccountId": "123456789012"
}
```

StartExecution

Contoh berikut menunjukkan catatan log CloudTrail yang menunjukkan tindakan `StartExecution`.

```
{
    "eventVersion": "1.04",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAJYDLDBVBI4EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/test-user",
        "accountId": "123456789012",
        "accessKeyId": "AKIAJL5C75K4ZEXAMPLE",
        "userName": "test-user"
    },
    "eventTime": "2016-10-28T01:17:25Z",
    "eventSource": "states.amazonaws.com",
    "eventName": "StartExecution",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "input": "{}",
        "stateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:testUser.2016-10-27-18-16-26.482bea32-560f-4a36-bd",
        "name": "testUser.2016-10-27-18-16-26.6e229586-3698-4ce5-8d"
    },
    "responseElements": {
        "startDate": "Oct 28, 2016 1:17:25 AM",
        "executionArn": "arn:aws:states:us-
east-1:123456789012:execution:testUser.2016-10-27-18-16-26.482bea32-560f-4a36-
bd:testUser.2016-10-27-18-16-26.6e229586-3698-4ce5-8d"
    },
    "requestID": "264c6f08-9cac-11e6-aed5-5b57d226e9ef",
    "eventID": "30a20c8e-a3a1-4b07-9139-cd9cd73b5eb8",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

StopExecution

Contoh berikut menunjukkan CloudTrail catatan log yang menunjukkan `StopExecution` tindakan.


```
{
    "eventVersion": "1.04",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAJYDLDBVBI4EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/test-user",
        "accountId": "123456789012",
        "accessKeyId": "AKIAJL5C75K4ZEXAMPLE",
        "userName": "test-user"
    },
    "eventTime": "2016-10-28T01:18:20Z",
    "eventSource": "states.amazonaws.com",
    "eventName": "StopExecution",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "executionArn": "arn:aws:states:us-
east-1:123456789012:execution:testUser.2016-10-27-18-17-00.337b3344-83:testUser.2016-10-27-18-1
    },
    "responseElements": {
        "stopDate": "Oct 28, 2016 1:18:20 AM"
    },
    "requestID": "4567625b-9cac-11e6-aed5-5b57d226e9ef",
    "eventID": "e658c743-c537-459a-aea7-dafb83c18c53",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```


Logging menggunakanCloudWatchLog

Alur Kerja Standar mencatat riwayat eksekusi diAWS Step Functions, meskipun Anda dapat mengkonfigurasi logging ke Amazon secara opsionalCloudWatchLog.

Tidak seperti Alur Kerja Standar, Alur Kerja Ekspres tidak mencatat riwayat eksekusi di AWS Step Functions. Untuk melihat riwayat eksekusi dan hasil Alur Kerja Ekspres, Anda harus mengonfigurasi pencatatan log ke AmazonCloudWatchLog. Penerbitan log tidak memblokir atau memperlambat eksekusi.

> **Note**
>
> Saat Anda mengkonfigurasi logging,CloudWatchBiaya logakan berlaku dan Anda akan ditagih pada tarif log vended. Untuk informasi lebih lanjut, lihatLog Vendingdi bawahLogtab padaCloudWatchHalaman harga.

Konfigurasikan log

Ketika Anda membuat alur kerja standar menggunakan konsol Step Functions, itu tidak akan dikonfigurasi untuk mengaktifkan logging keCloudWatchLog. Alur kerja Express yang dibuat menggunakan konsol Step Functions secara default akan dikonfigurasi untuk mengaktifkan logging keCloudWatchLog.

Untuk alur kerja Express, Step Functions dapat membuat peran dengan yang diperlukanAWS Identity and Access Management(IAM) kebijakan untukCloudWatchLog. Jika Anda membuat Alur Kerja Standar, atau Alur Kerja Ekspres menggunakan API, CLI, atau AWS CloudFormation, Step Functions tidak akan mengaktifkan log secara default, dan Anda perlu memastikan peran Anda memiliki izin yang diperlukan.

Untuk setiap eksekusi yang dimulai dari konsol, Step Functions menyediakan tautan keCloudWatchLog, dikonfigurasi dengan filter yang benar untuk mengambil peristiwa log khusus untuk eksekusi itu.

Untuk mengkonfigurasi logging, Anda dapat melewatiLoggingConfigurationparameter saat menggunakanCreateStateMachineatauUpdateStateMachine. Anda dapat menganalisis data Anda lebih lanjut diCloudWatchLog dengan menggunakanCloudWatchLog Wawasan. Untuk informasi lebih lanjut lihatMenganalisis Data Log denganCloudWatchWawasan Log.

CloudWatchLog muatan

Peristiwa riwayat eksekusi mungkin berisi properti input atau output dalam ketetapannya. Jika masukan lolos atau output lolos dikirim keCloudWatchLog melebihi 248KB, itu akan dipotong sebagai akibat dariCloudWatchKuota log.

- Anda dapat menentukan apakah muatan telah terpotong dengan meninjau properti `inputDetails` dan `outputDetails`. Untuk informasi selengkapnya, lihat `HistoryEventExecutionDataDetails` Tipe Data.

- Untuk Alur Kerja Standar, Anda dapat melihat riwayat eksekusi penuh dengan menggunakan `GetExecutionHistory`.
- `GetExecutionHistory` tidak tersedia untuk Alur Kerja Ekspres. Jika Anda ingin melihat input dan output penuh, Anda dapat menggunakan ARN Amazon S3. Untuk informasi selengkapnya, lihat the section called “Gunakan ARN Amazon S3 bukan meneruskan muatan besar”.

Kebijakan IAM untuk log keCloudWatchLog

Anda juga perlu mengonfigurasi peran IAM eksekusi mesin negara Anda agar memiliki izin yang tepat untuk masukCloudWatchLog seperti yang ditunjukkan pada contoh berikut.

Contoh kebijakan IAM

Berikut ini adalah contoh kebijakan yang dapat Anda gunakan untuk mengonfigurasi izin Anda. Seperti yang ditunjukkan pada contoh berikut, Anda perlu menentukan*di dalam`Resource`lapangan karenaCloudWatchTindakan API, sepertiCreateLogDeliverydanDescribeLogGroups, tidak mendukungJenis sumber daya yang ditentukan olehAmazon CloudWatch Logs. Untuk informasi lebih lanjut, lihatTindakan yang didefinisikan olehAmazon CloudWatch Logs.

- Untuk informasi tentangCloudWatchsumber daya, lihatCloudWatch Logssumber daya dan operasidi dalamAmazonCloudWatchPanduan Pengguna.
- Untuk informasi tentang izin yang Anda butuhkan untuk mengatur pengiriman log keCloudWatchLog, lihatIzin penggunadi bagian berjudulLog dikirim keCloudWatch Logs.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:CreateLogStream",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries",
                "logs:PutLogEvents",
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
```

```
                "logs:DescribeLogGroups"
            ],
            "Resource": "*"
        }
    ]
}
```

Tidak dapat mengaksesCloudWatchLog

Jika Anda tidak dapat mengaksesCloudWatchLog, pastikan Anda telah melakukan hal berikut:

1. Mengkonfigurasi peran IAM eksekusi mesin status Anda agar memiliki izin yang tepat untuk masukCloudWatchLog.

   Jika Anda menggunakanCreateStateMachineatauUpdateStateMachinepermintaan, pastikan Anda telah menentukan peran IAM di`roleArn`parameter yang berisi izin seperti yang ditunjukkan padacontoh sebelumnya.
2. MemeriksaCloudWatchKebijakan sumber daya log tidak melebihi batas karakter 5120 untukCloudWatchKebijakan sumber daya log.

   Jika Anda telah melampaui batas karakter, hapus izin yang tidak perlu dariCloudWatchKebijakan sumber daya log, atau awalan nama grup log dengan`/aws/vendedlogs`, yang akan memberikan izin ke grup log tanpa menambahkan lebih banyak karakter ke kebijakan sumber daya. Ketika Anda membuat grup log di konsol Step Functions, nama grup log diawali dengan `/aws/vendedlogs/states`. Untuk informasi selengkapnya, lihat Pembatasan ukuran kebijakan sumber daya Amazon CloudWatch Logs.

Tingkat Log

Anda dapat memilih dari `OFF`, `ALL`, `ERROR`, atau `FATAL`. Tidak ada log tipe peristiwa ketika diatur ke `OFF` dan semua jenis peristiwa lakukan ketika diatur ke `ALL`. Untuk `ERROR` dan `FATAL`, lihat tabel berikut.

Untuk informasi lebih lanjut tentang data eksekusi yang ditampilkan untuk eksekusi Express Workflow berdasarkan iniTingkat log, lihatEksekusi Alur Kerja Standar dan Ekspres di konsol.

Jenis Acara	ALL	ERROR	FATAL	OFF
ChoiceStateEntered	✓			
ChoiceStateExited	✓			
Execution Aborted	✓	✓	✓	
ExecutionFailed	✓	✓	✓	
ExecutionStarted	✓			
Execution Succeeded	✓			
Execution TimedOut	✓	✓	✓	
FailStateEntered	✓	✓		
LambdaFunctionFailed	✓	✓		
LambdaFunctionScheduled	✓			
LambdaFunctionScheduleFailed	✓	✓		
LambdaFunctionStarted	✓			
LambdaFunctionStartFailed	✓	✓		

Jenis Acara	ALL	ERROR	FATAL	OFF
LambdaFunctionSucceeded	✓			
LambdaFunctionTimedOut	✓	✓		
MapIterationAborted	✓	✓		
MapIterationFailed	✓	✓		
MapIterationStarted	✓			
MapIterationSucceeded	✓			
MapRunAborted	✓	✓		
MapRunFailed	✓	✓		
MapStateAborted	✓	✓		
MapStateEntered	✓			
MapStateExited	✓			
MapStateFailed	✓	✓		
MapStateStarted	✓			
MapStateSucceeded	✓			

Jenis Acara	ALL	ERROR	FATAL	OFF
ParallelStateAborted	✓	✓		
ParallelStateEntered	✓			
ParallelStateExited	✓			
ParallelStateFailed	✓	✓		
ParallelStateStarted	✓			
ParallelStateSucceeded	✓			
PassStateEntered	✓			
PassStateExited	✓			
SucceedStateEntered	✓			
SucceedStateExited	✓			
TaskFailed	✓	✓		
TaskScheduled	✓			
TaskStarted	✓			
TaskStartFailed	✓	✓		

Jenis Acara	ALL	ERROR	FATAL	OFF
TaskState Aborted	✓	✓		
TaskState Entered	✓			
TaskStateExited	✓			
TaskSubmitFailed	✓	✓		
TaskSubmitted	✓			
TaskSucceeded	✓			
TaskTimedOut	✓	✓		
WaitState Aborted	✓	✓		
WaitState Entered	✓			
WaitStateExited	✓			

AWS X-Ray dan Step Functions

Anda dapat menggunakan AWS X-Ray untuk memvisualisasikan komponen mesin status, mengidentifikasi hambatan performa, dan memecahkan masalah permintaan yang mengakibatkan kesalahan. Mesin status Anda mengirimkan data jejak ke X-Ray, dan X-Ray memproses data untuk menghasilkan peta layanan dan rangkuman jejak yang dapat dicari.

Dengan X-Ray diaktifkan untuk mesin status Anda, Anda dapat melacak permintaan saat dijalankan di Step Functions, di semua AWS Wilayah di mana X-Ray tersedia. Hal ini memberi Anda detail gambaran umum seluruh permintaan Step Functions. Step Functions akan mengirim jejak ke X-Ray untuk eksekusi mesin status, bahkan ketika ID penelusuran tidak diteruskan oleh layanan hulu. Anda dapat menggunakan peta layanan X-Ray untuk melihat latensi permintaan, termasuk AWS layanan

apa pun yang terintegrasi dengan X-Ray. Anda juga dapat mengonfigurasi aturan pengambilan sampel untuk memberi tahu X-Ray mengenai permintaan yang dicatat, tingkat pengambilan sampel, sesuai kriteria yang Anda tentukan.

Ketika X-Ray tidak diaktifkan untuk mesin status Anda, dan layanan hulu tidak meneruskan ID penelusuran, Step Functions tidak akan mengirim jejak ke X-Ray untuk eksekusi mesin status. Namun, jika ID penelusuran diteruskan oleh layanan hulu, Step Functions akan mengirim jejak ke X-Ray untuk eksekusi mesin status.

Anda dapat menggunakan AWS X-Ray Step Functions di wilayah di mana keduanya didukung. Lihat halaman titik akhir dan kuota Step Functionsdan X-Ray untuk informasi dukungan wilayah untuk X-Ray dan Step Functions.

Kuota Gabungan Fungsi X-Ray dan Step Functions

Anda dapat menambahkan data ke jejak hingga tujuh hari, dan data jejak kueri hingga tiga puluh hari sebelumnya, periode waktu yang disimpan oleh X-Ray. Jejak Anda akan dikenakan kuota X-Ray. Selain kuota lainnya, X-Ray menyediakan ukuran jejak minimal terjamin sebesar 100KB untuk mesin status Step Functions. Jika lebih dari 100KB data jejak diberikan ke X-Ray, hal ini dapat mengakibatkan penelusuran beku. Lihat bagian kuota layanan dari Titik akhir dan kuota X-Ray untuk informasi selengkapnya tentang kuota lain untuk X-Ray.

Important

Step Functions tidak mendukung penelusuran X-Ray untuk eksekusi alur kerja anak yang dimulai oleh status Peta Terdistribusi karena mudah untuk melampaui batas ukuran dokumen Trace untuk eksekusi tersebut.

Topik

- Penyiapan dan konfigurasi
- Konsep
- Integrasi layanan Step Functions dan X-Ray
- Melihat konsol X-Ray

- Melihat informasi pelacakan X-Ray untuk Step Functions
- Pelacakan
- Peta layanan
- Segmen dan subsegmen
- Analitik
- Konfigurasi
- Bagaimana jika tidak ada data dalam peta jejak atau peta layanan?

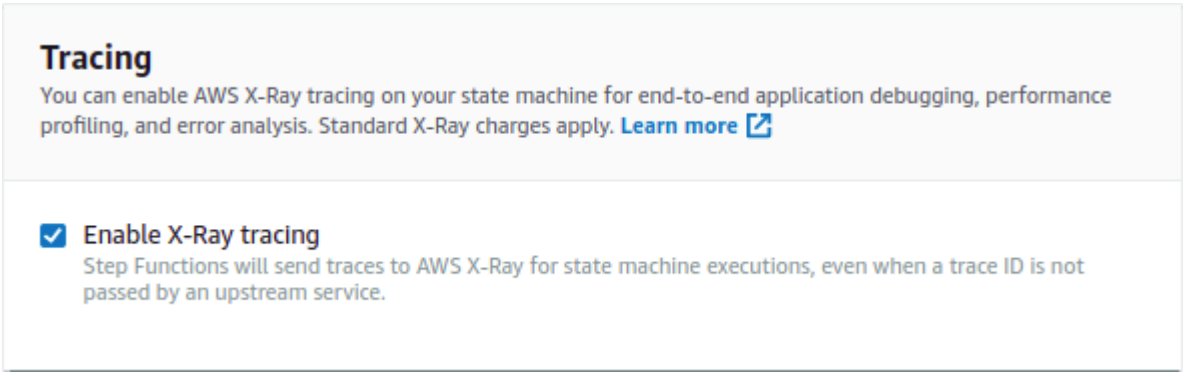
Penyiapan dan konfigurasi

Aktifkan penelusuran X-Ray saat membuat mesin status

Anda dapat mengaktifkan pelacakan X-Ray saat membuat mesin status baru dengan memilih Aktifkan penelusuran X-Ray pada halaman Tentukan detail.

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Pada halaman metode Choose authoring, pilih opsi yang sesuai untuk membuat mesin status Anda. Jika Anda memilih Jalankan proyek sampel, Anda tidak dapat mengaktifkan penelusuran X-Ray selama pembuatan mesin status, dan Anda harus mengaktifkan penelusuran X-Ray setelah mesin status Anda dibuat. Untuk informasi selengkapnya tentang mengaktifkan X-Ray di mesin status yang ada, lihatAktifkan penelusuran X-Ray di mesin status yang ada.

   Pilih Berikutnya.
3. Pada halaman Tentukan Detail, konfigurasikan mesin status Anda.
4. Pilih Aktifkan penelusuran X-Ray.



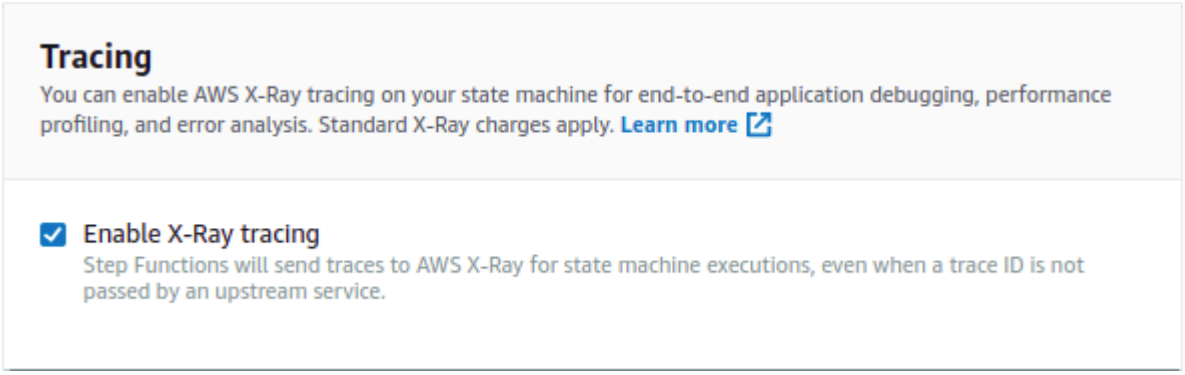
   Mesin status Step Functions Anda kini akan mengirim jejak ke X-Ray untuk eksekusi mesin status.

> **Note**
>
> Jika Anda memilih untuk menggunakan IAM role yang ada, Anda harus memastikan bahwa penulisan X-Ray diperbolehkan. Untuk informasi selengkapnya tentang izin yang Anda perlukan, lihat kebijakan IAM untuk X-Ray.

Aktifkan penelusuran X-Ray di mesin status yang ada

Untuk mengaktifkan penelusuran X-Ray di mesin status yang ada:

1. Di konsol Step Functions, pilih mesin status yang ingin Anda aktifkan pelacakan.
2. Pilih Edit.
3. Pilih Aktifkan penelusuran X-Ray.



Anda akan melihat notifikasi yang memberitahu Anda bahwa Anda mungkin perlu membuat perubahan tambahan.

> **Note**
>
> Jika Anda mengaktifkan X-Ray untuk mesin status yang ada, Anda harus memastikan bahwa Anda memiliki kebijakan IAM yang memberikan izin memadai bagi X-Ray untuk melakukan penelusuran. Anda dapat menambahkan satu secara manual, atau menghasilkan satu. Untuk informasi selengkapnya, lihat bagian kebijakan IAM untuk Kebijakan IAM untuk AWS X-Ray.

4. (Opsional) Buat peran baru secara otomatis bagi mesin status Anda untuk menyertakan izin X-Ray.

5. Pilih Simpan.

Mengonfigurasi penelusuran X-Ray untuk Step Functions

Saat pertama kali menjalankan mesin status dengan penelusuran X-Ray diaktifkan, itu akan menggunakan nilai konfigurasi default untuk penelusuran X-Ray. AWS X-Ray tidak mengumpulkan data untuk setiap permintaan yang dikirim ke aplikasi. Sebaliknya, ia mengumpulkan data untuk sejumlah permintaan yang signifikan secara statistik. Default mencatat permintaan pertama setiap detik dan lima persen dari permintaan tambahan. Satu permintaan per detik adalah waduk. Tindakan ini memastikan bahwa setidaknya satu pelacakan dicatat setiap detik selama layanan melayani permintaan. Lima persen adalah laju tempat permintaan tambahan di luar ukuran reservoir diambil sampelnya.

Untuk menghindari timbulnya biaya layanan ketika Anda memulai, hitungan sampling default adalah konservatif. Anda dapat mengonfigurasi X-Ray untuk mengubah aturan pengambilan sampel default dan mengonfigurasi aturan tambahan yang menerapkan pengambilan sampel berdasarkan properti layanan atau permintaan.

Misalnya, Anda mungkin ingin menonaktifkan sampling dan melacak semua permintaan panggilan yang mengubah status atau penanganan Akun AWS atau transaksi. Untuk panggilan baca-saja dengan volume tinggi, seperti poling latar belakang, pemeriksaan kondisi, atau pemeliharaan koneksi, Anda dapat mengalami kecepatan rendah dan masih mendapatkan data yang cukup untuk mengamati masalah yang terjadi.

Untuk mengonfigurasi aturan pengambilan sampel untuk mesin status Anda:

1. Buka konsol X-Ray.
2. Pilih Pengambilan sampel.
3. Untuk membuat aturan, pilih Buat aturan pengambilan sampel.

   Untuk mengedit aturan, pilih nama aturan.

   Untuk menghapus aturan, pilih aturan dan gunakan menu Tindakan untuk menghapusnya.

Beberapa bagian aturan pengambilan sampel yang ada, seperti nama dan prioritas, tidak dapat diubah. Sebagai gantinya, tambahkan atau klon aturan yang ada, buat perubahan yang Anda inginkan, lalu gunakan aturan baru.

Untuk detail informasi tentang aturan pengambilan sampel X-Ray dan cara mengonfigurasi berbagai parameter, lihat Mengonfigurasi aturan pengambilan sampel di konsol X-Ray.

Integrasikan layanan hulu

Untuk mengintegrasikan eksekusi alur kerja Step Functions, seperti alur kerja Express, Synchronous, dan Standard, dengan layanan upstream, Anda perlu menyetelnya. `traceHeader` Ini secara otomatis dilakukan untuk Anda jika Anda menggunakan API HTTP di API Gateway. Namun, jika Anda menggunakan fungsi Lambda dan/atau SDK, Anda perlu mengatur sendiri panggilan `traceHeader` on `StartExecution`atau `StartSyncExecution`API.

Anda harus menentukan `traceHeader` format sebagai`\p{ASCII}#`. Selain itu, untuk membiarkan Step Functions menggunakan ID jejak yang sama, Anda harus menentukan formatnya sebagai`Root={TRACE_ID};Sampled={1 or 0}`. Jika Anda menggunakan fungsi Lambda, ganti `TRACE_ID` dengan ID jejak di segmen Anda saat ini dan atur bidang Sampel seolah-olah mode pengambilan sampel Anda benar dan `1` `0` jika mode pengambilan sampel Anda salah. Menyediakan ID jejak dalam format ini memastikan bahwa Anda akan mendapatkan jejak lengkap.

Berikut ini adalah contoh yang ditulis dengan Python untuk menampilkan cara menentukan. `traceHeader`

```
state_machine = config.get_string_paramter("STATE_MACHINE_ARN")
    if (xray_recorder.current_subsegment() is not None and
        xray_recorder.current_subsegment().sampled) :
        trace_id = "Root={};Sampled=1".format(
            xray_recorder.current_subsegment().trace_id
        )
    else:
        trace_id = "Root=not enabled;Sampled=0"
    LOGGER.info("trace %s", trace_id)

    # execute it
    response = states.start_sync_execution(
        stateMachineArn=state_machine,
        input=event['body'],
        name=context.aws_request_id,
        traceHeader=trace_id
    )
    LOGGER.info(response)
```

Konsep

Konsol X-Ray

AWS X-Ray Konsol memungkinkan Anda untuk melihat peta layanan dan jejak untuk permintaan yang disajikan aplikasi Anda. Anda dapat mengakses konsol tersebut untuk melihat detail informasi yang dikumpulkan oleh X-Ray saat diaktifkan untuk mesin status Anda.

Lihat Melihat konsol X-Ray untuk informasi cara mengakses konsol X-Ray dengan tujuan mengeksekusi mesin status Anda.

Untuk detail informasi tentang konsol X-Ray, lihat Dokumentasi konsol X-Ray.

Segmen, subsegmen, dan jejak

Segmen mencatat informasi tentang permintaan ke mesin status Anda. Segmen berisi informasi seperti pekerjaan yang dilakukan mesin status Anda, dan mungkin juga berisi subsegmen dengan informasi tentang panggilan hilir.

Jejak mengumpulkan semua segmen yang dihasilkan oleh satu permintaan.

Pengambilan sampel

Untuk menjamin penelusuran yang efisien dan menyediakan sampel perwakilan permintaan yang dilayani aplikasi Anda, X-Ray menerapkan algoritme pengambilan sampel untuk memastikan permintaan yang ditelusuri. Hal ini dapat diubah dengan mengedit aturan pengambilan sampel.

Metrik

Untuk mesin status Anda, X-Ray akan mengukur waktu invokasi, waktu transisi status, waktu eksekusi keseluruhan Step Functions, dan keberagaman dalam waktu eksekusi ini. Informasi ini dapat diakses melalui konsol X-Ray.

Analitik

Konsol AWS X-Ray Analytics adalah alat interaktif untuk menafsirkan data jejak. Anda dapat mempersempit set data aktif dengan meningkatkan filter terperinci dengan mengeklik grafik dan panel metrik dan bidang yang terkait dengan rangkaian jejak saat ini. Hal ini memungkinkan Anda menganalisis performa mesin status Anda, dan dengan cepat menemukan serta mengidentifikasi masalah performa.

Untuk informasi rinci tentang analitik X-Ray, lihat Berinteraksi dengan konsol AWS X-Ray Analytics

Integrasi layanan Step Functions dan X-Ray

Beberapa AWS layanan yang terintegrasi dengan Step Functions menyediakan integrasi AWS X-Ray dengan menambahkan header tracing ke permintaan, menjalankan daemon X-Ray, atau membuat keputusan pengambilan sampel dan mengunggah data jejak ke X-Ray. Lainnya harus diinstrumentasi menggunakan AWS X-Ray SDK. Beberapa belum mendukung integrasi X-Ray. Integrasi X-Ray diperlukan untuk menyediakan data jejak yang lengkap saat menggunakan integrasi layanan dengan Step Functions

Dukungan X-Ray asli

Integrasi layanan dengan dukungan X-Ray asli meliputi:

- Layanan Pemberitahuan Sederhana Amazon
- Layanan Antrian Sederhana Amazon
- AWS Lambda
- AWS Step Functions

Instrumentasi diperlukan

Integrasi layanan yang memerlukan Instrumentasi X-Ray:

- Amazon Elastic Container Service
- AWS Batch
- AWS Fargate

Jejak sisi klien saja

Integrasi layanan lainnya tidak mendukung jejak X-Ray. Namun, jejak sisi klien masih dapat dikumpulkan:

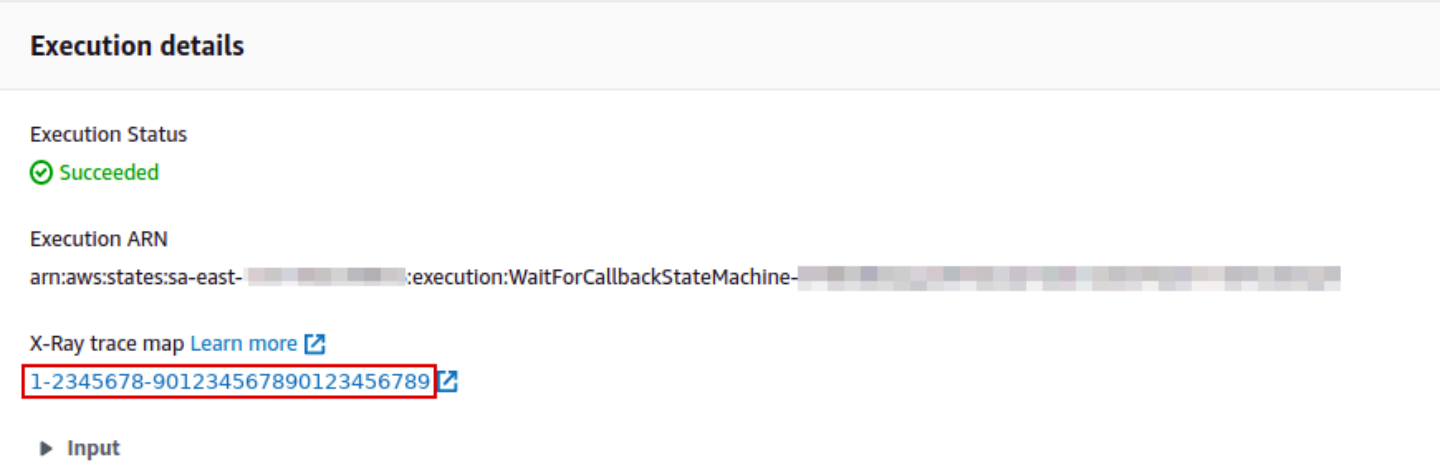
- Amazon DynamoDB
- Amazon EMR
- Amazon SageMaker

- AWS CodeBuild
- AWS Glue

Melihat konsol X-Ray

X-Ray menerima data dari layanan sebagai segmen. X-Ray mengelompokkan segmen yang memiliki permintaan umum ke penelusuran. X-Ray memproses jejak untuk menghasilkan grafik layanan yang menyediakan representasi visual dari aplikasi Anda.

Setelah Anda memulai eksekusi mesin status Anda, Anda dapat melihat jejak X-Ray dengan memilih tautan Peta jejak X-Ray di bagian Detail eksekusi.



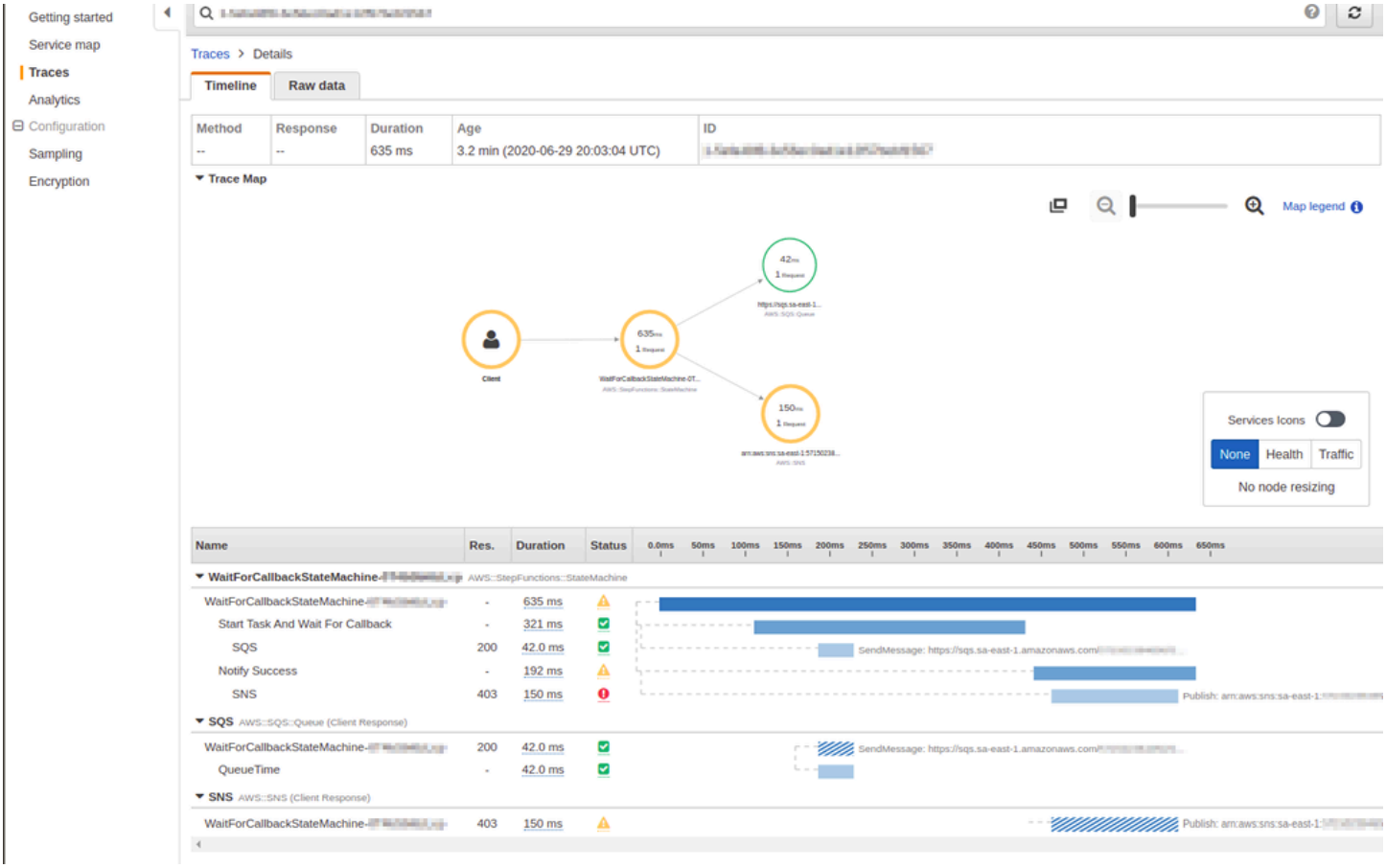
Setelah mengaktifkan X-Ray untuk mesin status, Anda dapat melihat informasi penelusuran untuk eksekusi di konsol X-Ray.

Melihat informasi pelacakan X-Ray untuk Step Functions

Langkah-langkah berikut menggambarkan tipe informasi yang dapat Anda lihat di konsol tersebut setelah Anda mengaktifkan X-Ray dan menjalankan eksekusi. Jejak X-Ray untuk proyek sampel Contoh Pola Panggilan Balik (Amazon SQS, Amazon SNS, Lambda) ditampilkan.

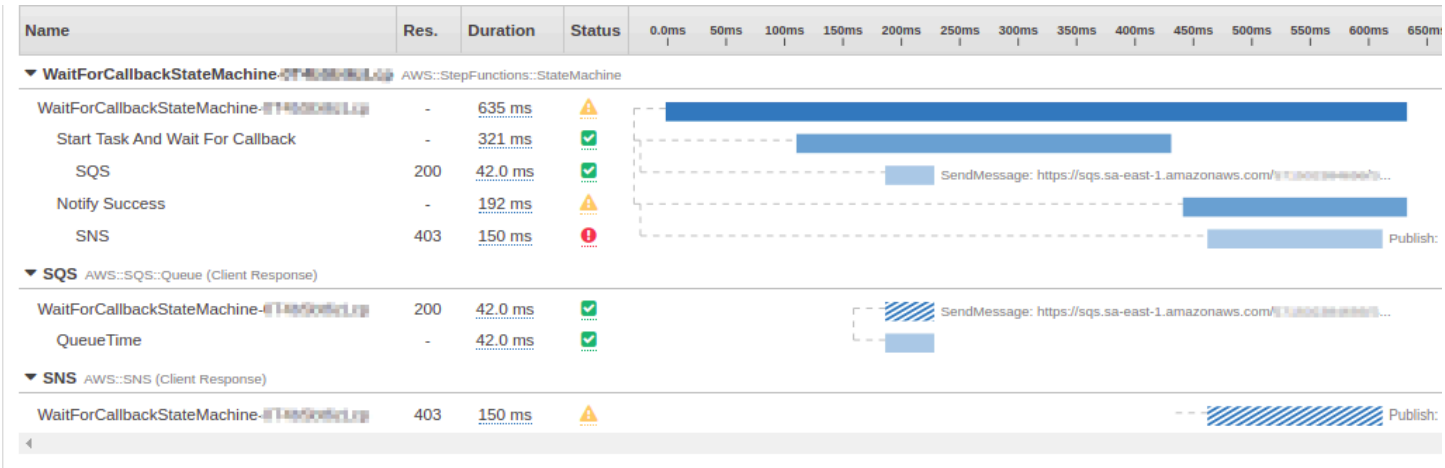
Pelacakan

Setelah eksekusi selesai, Anda dapat membuka konsol X-Ray, tempat Anda dapat melihat halaman Jejak X-Ray. Ini menampilkan gambaran umum peta layanan serta informasi jejak dan segmen untuk mesin status Anda.

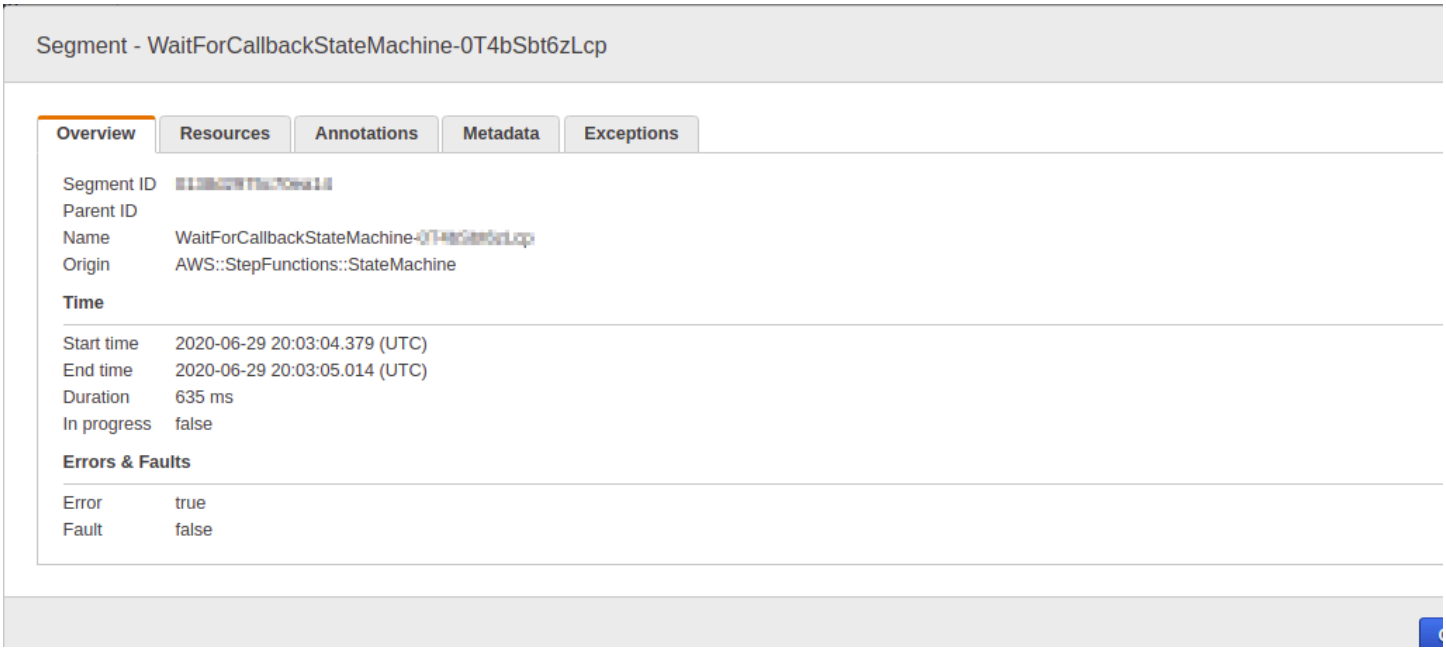


Peta layanan

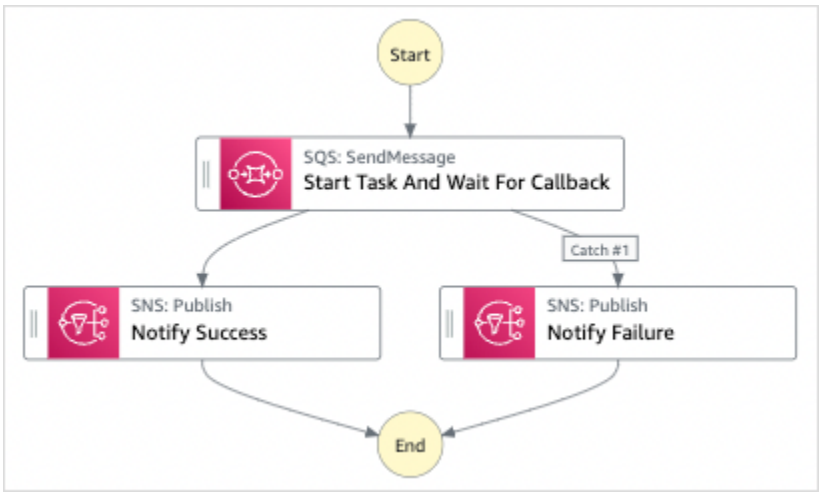
Peta layanan di konsol X-Ray membantu Anda mengidentifikasi layanan tempat kesalahan terjadi, tempat terjadinya koneksi dengan latensi tinggi, atau melihat jejak untuk permintaan yang tidak berhasil.



Pada peta jejak, Anda dapat memilih simpul layanan untuk melihat permintaan bagi simpul tersebut, atau tepi antara dua simpul untuk melihat permintaan yang melewati koneksi tersebut. Di sini, simpul `WaitForCallBack` telah dipilih, dan Anda dapat melihat informasi tambahan tentang eksekusi dan status respons.



Anda dapat melihat cara peta layanan X-Ray berkorelasi dengan mesin status. Terdapat simpul peta layanan untuk setiap integrasi layanan yang disebut dengan Step Functions, asalkan mendukung X-Ray.



Segmen dan subsegmen

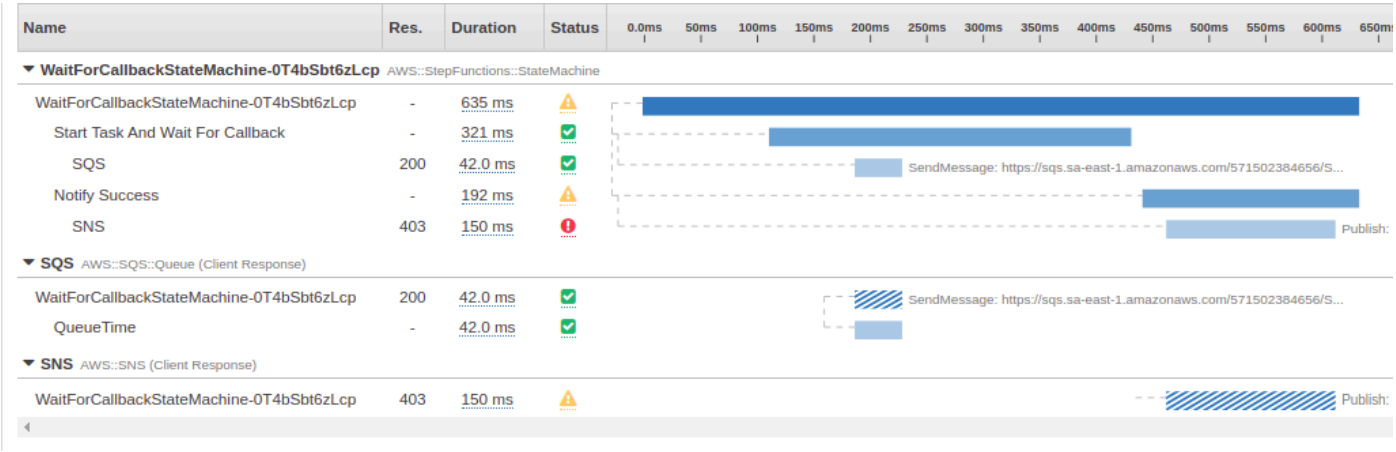
Jejak adalah kumpulan segmen yang dihasilkan oleh permintaan tunggal. Setiap segmen menyediakan nama sumber daya, detail permintaan, dan detail pekerjaan yang dilakukan. Di

halaman Jejak, Anda dapat melihat segmen dan jika diperluas, subsegmennya yang sesuai. Anda dapat memilih segmen atau subsegmen untuk melihat detail informasi hal tersebut.

Pilih setiap tab untuk melihat cara informasi untuk segmen dan subsegmen ditampilkan.

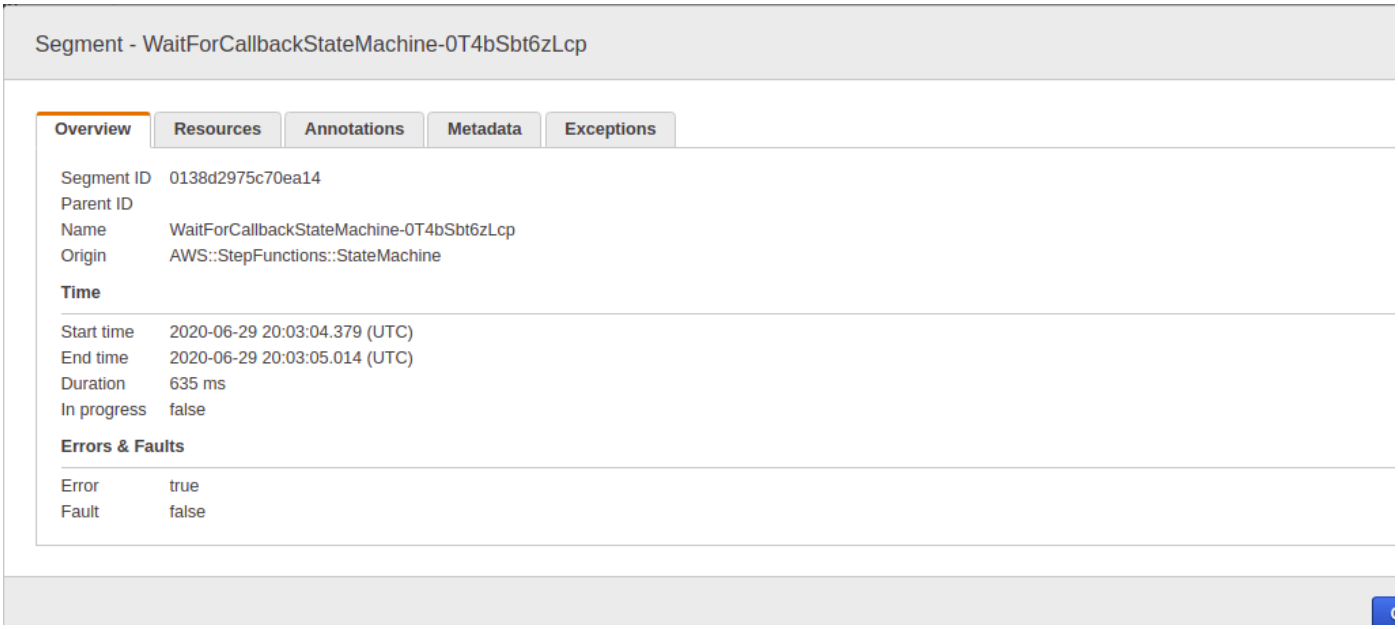
Overview of Segments

Gambaran umum segmen dan subsegmen untuk status ini. Terdapat segmen yang berbeda untuk setiap simpul pada peta layanan.



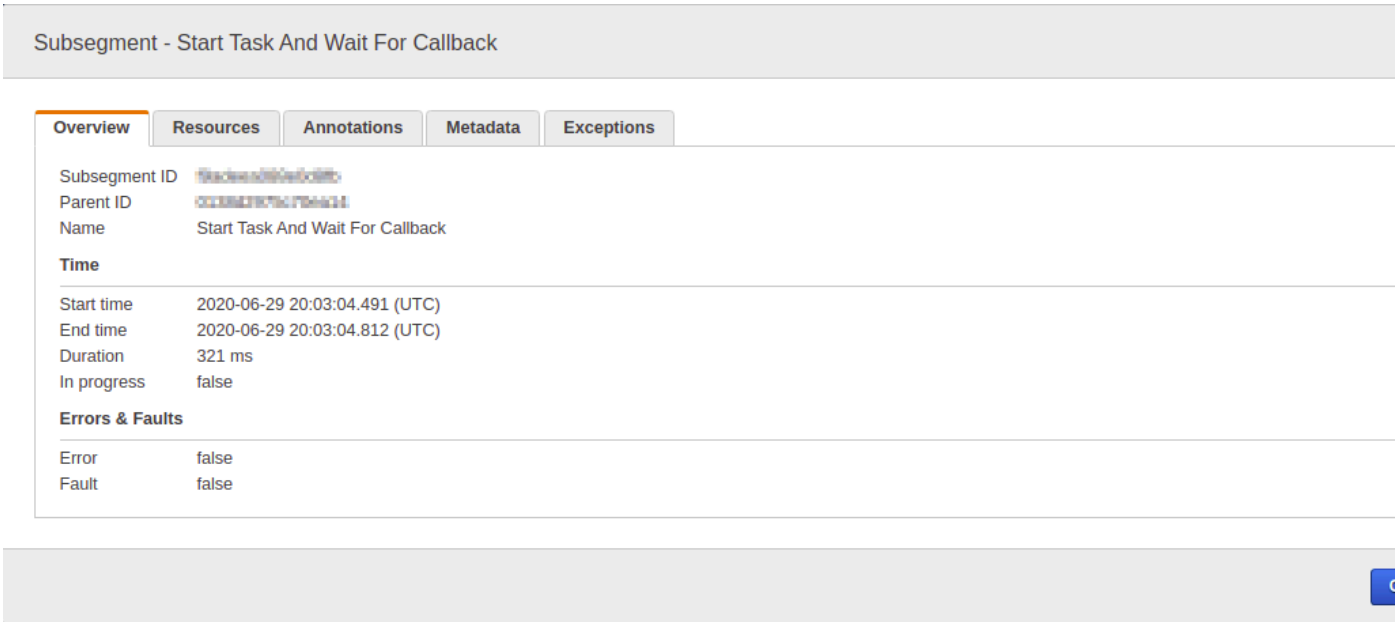
View segment detail

Memilih segmen menyediakan nama sumber daya, detail tentang permintaan dan pekerjaan yang dilakukan.



View subsegment detail

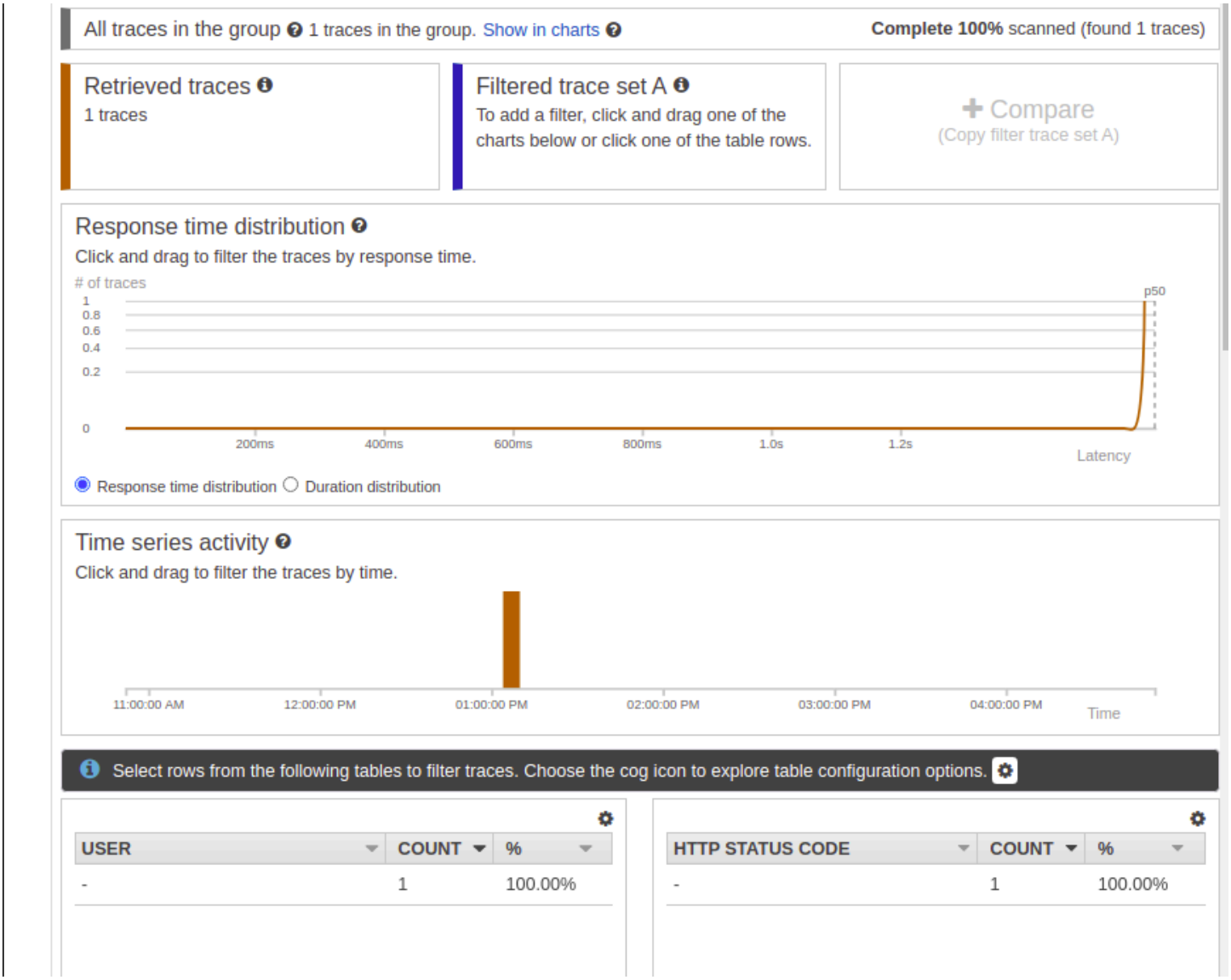
Segmen dapat memecah data tentang pekerjaan yang dilakukan menjadi subsegmen. Memilih subsegmen membuat Anda mampu melihat informasi dan detail waktu yang lebih terperinci. Subsegmen dapat berisi rincian tambahan tentang panggilan ke AWS layanan, API HTTP eksternal, atau database SQL.



Analitik

Konsol AWS X-Ray Analytics adalah alat interaktif untuk menafsirkan data jejak. Anda dapat menggunakan ini untuk memahami performa mesin status dengan lebih mudah. Konsol tersebut memungkinkan Anda mengeksplorasi, menganalisis, dan memvisualisasikan jejak melalui waktu respons interaktif dan grafik deret waktu. Hal ini dapat membantu Anda menemukan masalah performa dan latensi dengan cepat.

Anda dapat mempersempit set data aktif dengan meningkatkan filter terperinci dengan mengeklik grafik dan panel metrik dan bidang yang terkait dengan rangkaian jejak saat ini.

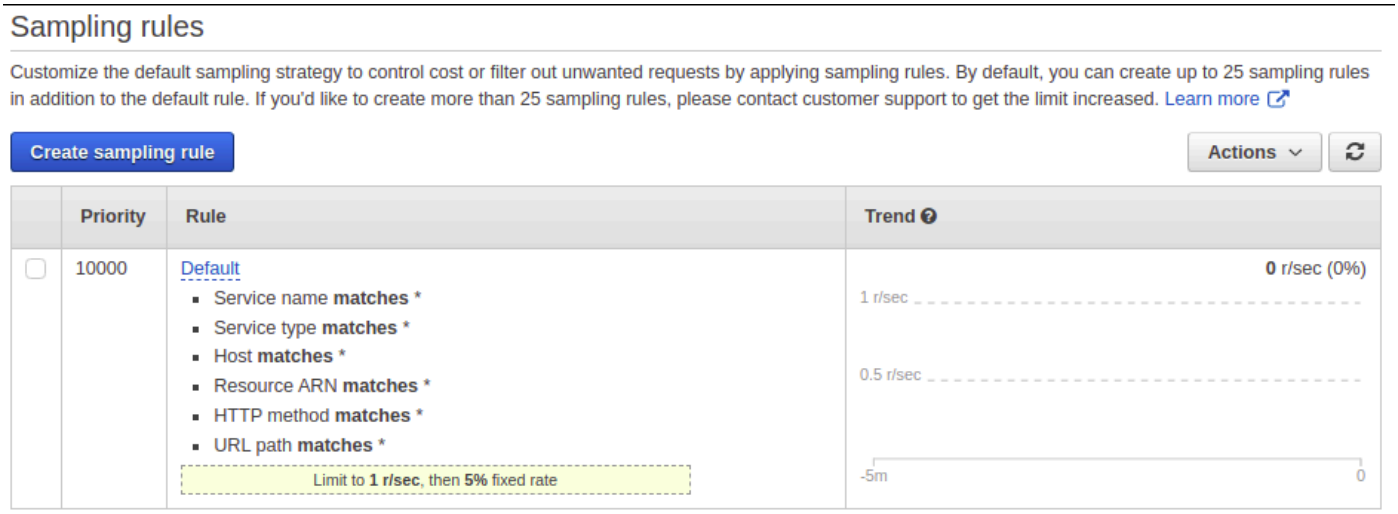


Konfigurasi

Anda dapat mengonfigurasi opsi pengambilan sampel dan enkripsi dari konsol X-Ray.

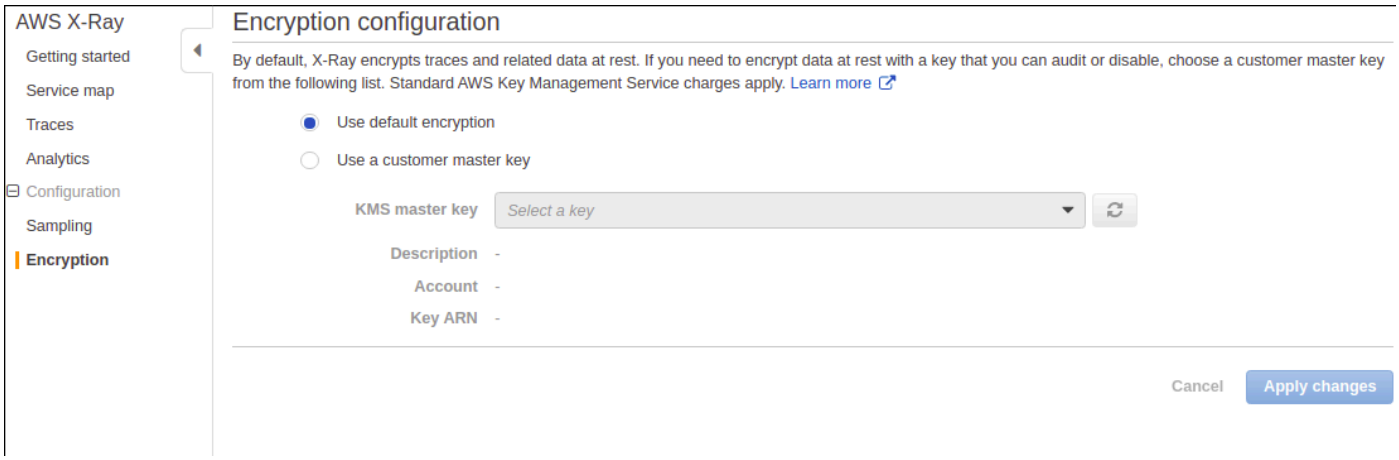
Sampling

Pilih Pengambilan sampel untuk melihat detail tingkat dan konfigurasi pengambilan sampel. Anda dapat mengubah aturan pengambilan sampel untuk mengontrol jumlah data yang Anda catat, dan mengubah perilaku pengambilan sampel sesuai dengan kebutuhan tertentu Anda.



Encryption

Pilih Enkripsi untuk mengubah pengaturan enkripsi. Anda dapat menggunakan pengaturan default, tempat X-Ray mengenkripsi jejak dan tanggal saat istirahat, atau, jika diperlukan, Anda dapat memilih kunci utama pelanggan. Biaya standar AWS KMS berlaku dalam kasus terakhir.



Bagaimana jika tidak ada data dalam peta jejak atau peta layanan?

Jika Anda telah mengaktifkan X-Ray, namun tidak dapat melihat data apa pun di konsol Sinar X, periksa apakah:

- IAM roles Anda diatur dengan benar untuk memungkinkan penulisan ke X-Ray.
- Aturan pengambilan sampel memungkinkan pengambilan sampel data.
- Karena dapat terjadi penundaan singkat sebelum penerapan IAM roles yang baru dibuat atau diubah, periksa jejak atau layanan peta lagi setelah beberapa menit.

- Jika Anda melihat Data Tidak Ditemukan di panel X-Ray Traces, periksa pengaturan akun IAM Anda dan pastikan itu AWS Security Token Service diaktifkan untuk wilayah yang dimaksud. Untuk informasi selengkapnya, lihat Mengaktifkan dan menonaktifkan AWS STS dalam Panduan Pengguna Wilayah AWS IAM.

Menggunakan Notifikasi Pengguna AWS dengan AWS Step Functions

Anda dapat menggunakan Notifikasi Pengguna AWSuntuk mengatur saluran pengiriman untuk mendapatkan pemberitahuan tentang AWS Step Functions acara. Anda akan menerima sebuah notifikasi saat ada sebuah peristiwa yang cocok dengan sebuah aturan yang Anda tentukan. Anda dapat menerima pemberitahuan untuk acara melalui beberapa saluran, termasuk email, pemberitahuan AWS Chatbotobrolan, atau pemberitahuan AWS Console Mobile Applicationpush. Anda juga dapat melihat notifikasi di Pusat Pemberitahuan Konsol. Notifikasi Penggunamendukung agregasi, yang dapat mengurangi jumlah notifikasi yang Anda terima selama acara tertentu.

Keamanan di AWS Step Functions

Bagian ini memberikan informasi tentang AWS Step Functions keamanan dan otentikasi.

Step Functions menggunakan IAM untuk mengontrol akses ke AWS layanan dan sumber daya lain. Untuk gambaran umum mengenai cara kerja IAM, lihat Gambaran Umum Manajemen Akses di Panduan Pengguna IAM. Untuk ikhtisar kredenal keamanan, lihat Kredenal AWS Keamanan di. Referensi Umum Amazon Web

Perlindungan data di AWS Step Functions

Model tanggung jawab AWS bersama model berlaku untuk perlindungan data di AWS Step Functions. 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 sumber daya. AWS Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya Layanan AWS.
- Gunakan layanan keamanan terkelola lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.

- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-2 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi yang lebih lengkap tentang titik akhir FIPS yang tersedia di Standar Pemrosesan Informasi Federal (FIPS) 140-2.

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 Step Functions atau lainnya Layanan AWS menggunakan konsol, API AWS CLI, atau AWS SDK. 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.

Enkripsi di AWS Step Functions

Enkripsi saat Data Tidak Berpindah

Step Functions selalu mengenkripsi data at rest Anda. Data dalam AWS Step Functions dienkripsi saat istirahat menggunakan enkripsi sisi server transparan. Hal ini membantu mengurangi beban operasional dan kompleksitas yang terlibat dalam melindungi data sensitif. Dengan enkripsi at rest, Anda dapat membangun aplikasi yang sensitif terhadap keamanan yang memenuhi persyaratan kepatuhan enkripsi dan peraturan

Enkripsi dalam transit

Step Functions mengenkripsi data dalam transit antara layanan dan layanan AWS terintegrasi lainnya (lihat Menggunakan AWS Step Functions dengan layanan lain). Semua data yang melewati antara Step Functions dan layanan terintegrasi dienkripsi menggunakan Keamanan Lapisan Pengangkutan (TLS).

Manajemen Identitas dan Akses di AWS Step Functions

Akses ke AWS Step Functions memerlukan kredensyal yang AWS dapat digunakan untuk mengautentikasi permintaan Anda. Kredensil ini harus memiliki izin untuk mengakses AWS sumber daya, seperti mengambil data peristiwa dari sumber daya lain. AWS Bagian berikut memberikan rincian tentang bagaimana Anda dapat menggunakan AWS Identity and Access Management (IAM)

dan Step Functions untuk membantu mengamankan sumber daya Anda dengan mengontrol siapa yang dapat mengaksesnya.

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 Step Functions. IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan di Step Functions.

Pengguna layanan - Jika Anda menggunakan layanan Step Functions untuk melakukan pekerjaan Anda, administrator Anda memberi Anda kredensyal dan izin yang Anda butuhkan. Saat Anda menggunakan lebih banyak fitur Step Functions untuk melakukan pekerjaan Anda, Anda mungkin memerlukan izin tambahan. Memahami cara akses dikelola dapat membantu Anda meminta izin yang tepat dari administrator Anda. Jika Anda tidak dapat mengakses fitur di Step Functions, lihatMemecahkan masalah AWS Step Functions identitas dan akses.

Administrator layanan - Jika Anda bertanggung jawab atas sumber daya Step Functions di perusahaan Anda, Anda mungkin memiliki akses penuh ke Step Functions. Tugas Anda adalah menentukan fitur dan sumber daya Step Functions mana yang harus diakses pengguna layanan Anda. Kemudian, Anda harus mengirimkan permintaan kepada administrator IAM Anda untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep Basic IAM. Untuk mempelajari lebih lanjut tentang bagaimana perusahaan Anda dapat menggunakan IAM dengan Step Functions, lihatBagaimana AWS Step Functions bekerja dengan IAM.

Administrator IAM - Jika Anda administrator IAM, Anda mungkin ingin mempelajari detail tentang cara menulis kebijakan untuk mengelola akses ke Step Functions. Untuk melihat contoh kebijakan berbasis identitas Step Functions yang dapat Anda gunakan di IAM, lihat. Contoh kebijakan berbasis identitas untuk AWS Step Functions

Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensi identitas Anda. Anda harus diautentikasi (masuk ke AWS) sebagai Pengguna root akun AWS, sebagai pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk AWS sebagai identitas federasi dengan menggunakan kredensil yang disediakan melalui sumber identitas. AWS IAM Identity Center Pengguna (IAM Identity Center), autentikasi masuk tunggal perusahaan Anda, dan kredensyal Google atau Facebook Anda adalah contoh identitas federasi. Saat Anda masuk sebagai identitas gabungan, administrator Anda sebelumnya menyiapkan federasi identitas menggunakan peran IAM. Ketika Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil peran.

Bergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau portal AWS akses. Untuk informasi selengkapnya tentang masuk AWS, lihat Cara masuk ke Panduan AWS Sign-In Pengguna Anda Akun AWS.

Jika Anda mengakses AWS secara terprogram, AWS sediakan kit pengembangan perangkat lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara kriptografis dengan menggunakan kredensyal Anda. Jika Anda tidak menggunakan AWS alat, Anda harus menandatangani permintaan sendiri. Untuk informasi selengkapnya tentang penggunaan metode yang disarankan untuk menandatangani permintaan sendiri, lihat Menandatangani permintaan AWS API di Panduan Pengguna IAM.

Apa pun metode autentikasi yang digunakan, Anda mungkin diminta untuk menyediakan informasi keamanan tambahan. Misalnya, AWS merekomendasikan agar Anda menggunakan otentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun Anda. Untuk mempelajari selengkapnya, silakan lihat Autentikasi multi-faktor dalam Panduan Pengguna AWS IAM Identity Center dan Menggunakan autentikasi multi-faktor (MFA) di AWS dalam Panduan Pengguna IAM.

Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya di akun. Identitas ini disebut pengguna Akun AWS root dan diakses dengan masuk dengan alamat email dan kata sandi yang Anda gunakan untuk membuat akun. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Lindungi kredensial pengguna root Anda dan gunakan kredensial tersebut untuk melakukan tugas yang hanya dapat dilakukan pengguna root. Untuk daftar lengkap tugas yang mengharuskan Anda masuk sebagai pengguna root, lihat bagian Tugas yang memerlukan kredensial pengguna root dalam Panduan Pengguna IAM.

Identitas gabungan

Sebagai praktik terbaik, mewajibkan pengguna manusia, termasuk pengguna yang memerlukan akses administrator, untuk menggunakan federasi dengan penyedia identitas untuk mengakses Layanan AWS dengan menggunakan kredensyal sementara.

Identitas federasi adalah pengguna dari direktori pengguna perusahaan Anda, penyedia identitas web, direktori Pusat Identitas AWS Directory Service, atau pengguna mana pun yang mengakses Layanan AWS dengan menggunakan kredensyal yang disediakan melalui sumber identitas. Ketika identitas federasi mengakses Akun AWS, mereka mengambil peran, dan peran memberikan kredensi sementara.

Untuk manajemen akses terpusat, kami merekomendasikan penggunaan AWS IAM Identity Center. Anda dapat membuat pengguna dan grup di Pusat Identitas IAM, atau Anda dapat menghubungkan dan menyinkronkan ke sekumpulan pengguna dan grup di sumber identitas Anda sendiri untuk digunakan di semua aplikasi Akun AWS dan aplikasi Anda. Untuk informasi tentang Pusat Identitas IAM, lihat Apakah yang dimaksud Pusat Identitas IAM? dalam Panduan Pengguna AWS IAM Identity Center .

Pengguna dan grup IAM

Pengguna IAM adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus untuk satu orang atau aplikasi. Apabila memungkinkan, kami merekomendasikan untuk mengandalkan pada kredensial sementara alih-alih membuat pengguna IAM yang memiliki kredensial jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan tertentu yang memerlukan kredensial jangka panjang dengan pengguna IAM, kami merekomendasikan Anda merotasi kunci akses. Untuk informasi selengkapnya, silakan lihat Merotasi kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensial jangka panjang dalam Panduan Pengguna IAM.

Grup IAM adalah identitas yang menentukan sekumpulan pengguna IAM. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin bagi beberapa pengguna sekaligus. Grup mempermudah pengelolaan izin untuk sejumlah besar pengguna sekaligus. Sebagai contoh, Anda dapat memiliki grup yang diberi nama IAMAdmins dan memberikan izin kepada grup tersebut untuk mengelola sumber daya IAM.

Pengguna berbeda dari peran. Pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran dimaksudkan untuk dapat digunakan oleh siapa pun yang membutuhkannya. Pengguna memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk

mempelajari selengkapnya, silakan lihat Kapan harus membuat pengguna IAM (bukan peran) dalam Panduan Pengguna IAM.

Peran IAM

Peran IAM adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus. Peran ini mirip dengan pengguna IAM, tetapi tidak terkait dengan orang tertentu. Anda dapat mengambil peran IAM untuk sementara AWS Management Console dengan beralih peran. Anda dapat mengambil peran dengan memanggil operasi AWS CLI atau AWS API atau dengan menggunakan URL kustom. Untuk informasi selengkapnya tentang cara menggunakan peran, lihat Menggunakan peran IAM dalam Panduan Pengguna IAM.

Peran IAM dengan kredensial sementara berguna dalam situasi berikut:

- Akses pengguna gabungan – Untuk menetapkan izin ke sebuah identitas gabungan, Anda dapat membuat peran dan menentukan izin untuk peran tersebut. Ketika identitas gabungan mengautentikasi, identitas tersebut terhubung dengan peran dan memperoleh izin yang ditentukan oleh peran. Untuk informasi tentang peran-peran untuk federasi, lihat Membuat peran untuk Penyedia Identitas pihak ketiga dalam Panduan Pengguna IAM. Jika menggunakan Pusat Identitas IAM, Anda harus mengonfigurasi serangkain izin. Untuk mengontrol apa yang dapat diakses oleh identitas Anda setelah diautentikasi, Pusat Identitas IAM menghubungkan izin yang ditetapkan dengan peran di IAM. Untuk informasi tentang rangkaian izin, lihat Rangkaian izin dalam Panduan Pengguna AWS IAM Identity Center .
- Izin pengguna IAM sementara – Pengguna atau peran IAM dapat mengambil peran IAM guna mendapatkan berbagai izin secara sementara untuk tugas tertentu.
- Akses lintas akun – Anda dapat menggunakan peran IAM untuk mengizinkan seseorang (pengguna utama tepercaya) dengan akun berbeda untuk mengakses sumber daya yang ada di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, dengan beberapa Layanan AWS, Anda dapat melampirkan kebijakan secara langsung ke sumber daya (alih-alih menggunakan peran sebagai proxy). Untuk mempelajari perbedaan antara kebijakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Bagaimana peran IAM berbeda dari kebijakan berbasis sumber daya dalam Panduan Pengguna IAM.
- Akses lintas layanan — Beberapa Layanan AWS menggunakan fitur lain Layanan AWS. Sebagai contoh, ketika Anda memanggil suatu layanan, biasanya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Layanan dapat melakukan hal tersebut menggunakan izin pengguna utama, menggunakan peran layanan, atau menggunakan peran terkait layanan.

- Sesi akses teruskan (FAS) — Saat Anda menggunakan pengguna atau peran IAM untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Saat menggunakan beberapa layanan, Anda mungkin melakukan tindakan yang kemudian dilanjutkan oleh tindakan lain pada layanan yang berbeda. FAS menggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain Layanan AWS atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat Sesi akses maju.
- Peran layanan – Peran layanan adalah peran IAM yang dijalankan oleh layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, memodifikasi, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan Pengguna IAM.
- Peran terkait layanan — Peran terkait layanan adalah jenis peran layanan yang ditautkan ke peran layanan. Layanan AWS Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.

- Aplikasi yang berjalan di Amazon EC2 — Anda dapat menggunakan peran IAM untuk mengelola kredensyal sementara untuk aplikasi yang berjalan pada instans EC2 dan membuat atau permintaan API. AWS CLI AWS Cara ini lebih dianjurkan daripada menyimpan kunci akses dalam instans EC2. Untuk menetapkan AWS peran ke instans EC2 dan membuatnya tersedia untuk semua aplikasinya, Anda membuat profil instance yang dilampirkan ke instance. Profil instans berisi peran dan memungkinkan program yang berjalan pada instans EC2 untuk mendapatkan kredensial sementara. Untuk informasi selengkapnya, lihat Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan di instans Amazon EC2 dalam Panduan Pengguna IAM.

Untuk mempelajari apakah kita harus menggunakan peran IAM atau pengguna IAM, lihat Kapan harus membuat peran IAM (bukan pengguna) dalam Panduan Pengguna IAM.

Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan adalah objek AWS yang, ketika dikaitkan dengan identitas atau sumber daya, menentukan izinnya. AWS mengevaluasi kebijakan ini ketika prinsipal (pengguna, pengguna root, atau sesi peran) membuat permintaan. Izin dalam kebijakan dapat menentukan permintaan

yang diizinkan atau ditolak. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang struktur dan isi dokumen kebijakan JSON, silakan lihat Gambaran Umum kebijakan JSON dalam Panduan Pengguna IAM.

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, pengguna utama manakah yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Untuk memberikan izin kepada pengguna untuk melakukan tindakan pada sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat menjalankan peran.

Kebijakan IAM mendefinisikan izin untuk suatu tindakan terlepas dari metode yang Anda gunakan untuk operasi. Sebagai contoh, anggap saja Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole`. Pengguna dengan kebijakan tersebut bisa mendapatkan informasi peran dari AWS Management Console, API AWS CLI, atau AWS API.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol tindakan apa yang dapat dilakukan oleh pengguna dan peran, pada sumber daya mana, dan dalam keadaan apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan terkelola. Kebijakan inline disematkan langsung ke satu pengguna, grup, atau peran. Kebijakan terkelola adalah kebijakan mandiri yang dapat Anda lampirkan ke beberapa pengguna, grup, dan peran dalam. Akun AWS Kebijakan AWS terkelola mencakup kebijakan terkelola dan kebijakan yang dikelola pelanggan. Untuk mempelajari cara memilih antara kebijakan terkelola atau kebijakan inline, lihat Memilih 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. Contoh kebijakan berbasis sumber daya adalah kebijakan tepercaya 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. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan

oleh pengguna utama tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan pengguna utama dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

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.

Daftar kontrol akses (ACL)

Daftar kontrol akses (ACL) mengendalikan pengguna utama mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL serupa dengan kebijakan berbasis sumber daya, meskipun tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung ACL. Untuk mempelajari ACL selengkapnya, lihat Gambaran umum daftar kontrol akses (ACL) dalam Panduan Pengembang Amazon Simple Storage Service.

Jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang kurang umum. Jenis-jenis kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda berdasarkan jenis kebijakan yang lebih umum.

- Batasan izin – Batasan izin adalah fitur lanjutan di mana Anda menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM (pengguna atau peran IAM). Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas milik entitas dan batasan izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan secara eksplisit terhadap salah satu kebijakan ini akan mengesampingkan izin tersebut. Untuk informasi selengkapnya tentang batasan izin, lihat Batasan izin untuk entitas IAM dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCP) — SCP adalah kebijakan JSON yang menentukan izin maksimum untuk organisasi atau unit organisasi (OU) di. AWS Organizations AWS Organizations adalah layanan untuk mengelompokkan dan mengelola secara terpusat beberapa Akun AWS yang dimiliki bisnis Anda. Jika Anda mengaktifkan semua fitur di organisasi, Anda dapat menerapkan kebijakan kontrol layanan (SCP) ke salah satu atau semua akun Anda. SCP membatasi izin untuk entitas di akun anggota, termasuk masing-masing. Pengguna root akun AWS Untuk informasi selengkapnya tentang Organisasi dan SCP, lihat Cara kerja SCP dalam Panduan Pengguna AWS Organizations .
- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda teruskan sebagai parameter saat Anda membuat sesi sementara secara terprogram untuk peran atau pengguna gabungan. Izin

sesi yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga bisa datang dari kebijakan berbasis sumber daya. Penolakan secara eksplisit terhadap salah satu kebijakan ini akan mengesampingkan izin tersebut. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

Jika beberapa jenis kebijakan diterapkan pada suatu permintaan, izin yang dihasilkan akan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat Logika evaluasi kebijakan di Panduan Pengguna IAM.

Kontrol Akses

Anda dapat memiliki kredensial yang valid untuk melakukan autentikasi permintaan, tetapi kecuali jika Anda memiliki izin, Anda tidak dapat membuat atau mengakses sumber daya Step Functions. Misalnya, Anda harus memiliki izin untuk memanggil, Amazon Simple Notification Service (AWS Lambda Amazon SNS), dan Amazon Simple Queue Service (Amazon SQS) menargetkan target yang terkait dengan aturan Step Functions Anda.

Bagian berikut menggambarkan cara mengelola izin untuk Step Functions.

- Membuat peran IAM untuk mesin negara Anda
- Membuat Izin IAM Terperinci untuk Pengguna Non-Admin
- Amazon VPC Endpoints untuk Step Functions
- Kebijakan IAM untuk layanan terintegrasi
- Kebijakan IAM untuk menggunakan status Peta Terdistribusi

Tindakan kebijakan untuk Step Functions

Mendukung tindakan kebijakan	Ya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, pengguna utama manakah yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan-tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Tindakan kebijakan biasanya memiliki nama yang sama dengan operasi AWS API terkait. Ada beberapa pengecualian, misalnya tindakan hanya izin yang tidak memiliki operasi API yang cocok. Ada juga beberapa operasi yang memerlukan beberapa tindakan dalam suatu kebijakan. Tindakan tambahan ini disebut tindakan dependen.

Menyertakan tindakan dalam kebijakan untuk memberikan izin untuk melakukan operasi terkait.

Untuk melihat daftar tindakan Step Functions, lihat Sumber Daya yang Ditentukan oleh AWS Step Functions dalam Referensi Otorisasi Layanan.

Tindakan kebijakan di Step Functions menggunakan awalan berikut sebelum tindakan:

```
states
```

Untuk menetapkan secara spesifik beberapa tindakan dalam satu pernyataan, pisahkan tindakan-tindakan tersebut dengan koma.

```
"Action": [
      "states:action1",
      "states:action2"
         ]
```

Untuk melihat contoh kebijakan berbasis identitas Step Functions, lihat. Contoh kebijakan berbasis identitas untuk AWS Step Functions

Sumber daya kebijakan untuk Step Functions

Mendukung sumber daya kebijakan	Ya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, pengguna utama manakah yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Elemen kebijakan JSON `Resource` menentukan objek yang menjadi target penerapan tindakan. Pernyataan harus menyertakan entah elemen `Resource` atau `NotResource`. Praktik terbaiknya,

tentukan sumber daya menggunakan Amazon Resource Name (ARN). Anda dapat melakukan ini untuk tindakan yang mendukung jenis sumber daya tertentu, yang dikenal sebagai izin tingkat sumber daya.

Untuk tindakan yang tidak mendukung izin di tingkat sumber daya, misalnya operasi pencantuman, gunakan wildcard (*) untuk menunjukkan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

```
"Resource": "*"
```

Untuk melihat daftar tipe sumber daya Step Functions dan ARNnya, lihat Tindakan yang Ditentukan oleh AWS Step Functions dalam Referensi Otorisasi Layanan. Untuk mempelajari tindakan mana yang dapat Anda tentukan ARN dari setiap sumber daya, lihat Sumber Daya yang Ditentukan oleh. AWS Step Functions

Untuk melihat contoh kebijakan berbasis identitas Step Functions, lihat. Contoh kebijakan berbasis identitas untuk AWS Step Functions

Kunci kondisi kebijakan untuk Step Functions

Mendukung kunci kondisi kebijakan spesifik layanan	Ya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, pengguna utama manakah yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Elemen `Condition` (atau blok `Condition`) memungkinkan Anda menentukan kondisi di mana suatu pernyataan akan diterapkan. Elemen `Condition` bersifat opsional. Anda dapat membuat ekspresi kondisional yang menggunakan operator kondisi, misalnya sama dengan atau kurang dari, untuk mencocokkan kondisi dalam kebijakan dengan nilai-nilai yang diminta.

Jika Anda menentukan beberapa elemen `Condition` dalam sebuah pernyataan, atau beberapa kunci dalam elemen `Condition` tunggal, AWS akan mengevaluasinya dengan menggunakan operasi `AND` yang logis. Jika Anda menentukan beberapa nilai untuk satu kunci kondisi, AWS mengevaluasi kondisi menggunakan `OR` operasi logis. Semua kondisi harus dipenuhi sebelum izin pernyataan diberikan.

Anda juga dapat menggunakan variabel placeholder saat menentukan kondisi. Sebagai contoh, Anda dapat memberikan izin kepada pengguna IAM untuk mengakses sumber daya hanya jika izin tersebut mempunyai tanda yang sesuai dengan nama pengguna IAM mereka. Untuk informasi selengkapnya, silakan lihat Elemen kebijakan IAM: variabel dan tanda dalam Panduan Pengguna IAM.

AWS mendukung kunci kondisi global dan kunci kondisi khusus layanan. Untuk melihat semua kunci kondisi AWS global, lihat kunci konteks kondisi AWS global di Panduan Pengguna IAM.

Untuk melihat daftar kunci kondisi Step Functions, lihat Condition Keys untuk AWS Step Functions Referensi Otorisasi Layanan. Untuk mempelajari tindakan dan sumber daya yang dapat Anda gunakan kunci kondisi, lihat Sumber Daya yang Ditentukan oleh AWS Step Functions.

Untuk melihat contoh kebijakan berbasis identitas Step Functions, lihat. Contoh kebijakan berbasis identitas untuk AWS Step Functions

ACL dalam Step Functions

Mendukung ACL	Tidak

Daftar kontrol akses (ACL) mengendalikan pengguna utama mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL serupa dengan kebijakan berbasis sumber daya, meskipun tidak menggunakan format dokumen kebijakan JSON.

ABAC dengan Step Functions

Mendukung ABAC (tanda dalam kebijakan)	Parsial

Kontrol akses berbasis atribut (ABAC) adalah strategi otorisasi yang menentukan izin berdasarkan atribut. Dalam AWS, atribut ini disebut tag. Anda dapat melampirkan tag ke entitas IAM (pengguna atau peran) dan ke banyak AWS sumber daya. Penandaan ke entitas dan sumber daya adalah langkah pertama dari ABAC. Kemudian, rancanglah kebijakan ABAC untuk mengizinkan operasi-operasi ketika tanda milik pengguna utama cocok dengan tanda yang ada di sumber daya yang ingin diakses.

ABAC sangat berguna di lingkungan yang berkembang dengan cepat dan berguna di situasi di mana pengelolaan kebijakan menjadi rumit.

Untuk mengendalikan akses berdasarkan tanda, berikan informasi tentang tanda di elemen syarat dari sebuh kebijakan dengan menggunakan kunci-kunci persyaratan `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, atau `aws:TagKeys`.

Jika sebuah layanan mendukung ketiga kunci persyaratan untuk setiap jenis sumber daya, nilainya adalah Ya untuk layanan tersebut. Jika suatu layanan mendukung ketiga kunci kondisi hanya untuk beberapa jenis sumber daya, nilainya adalah Parsial.

Untuk informasi selengkapnya tentang ABAC, lihat Apa itu ABAC? dalam Panduan Pengguna IAM. Untuk melihat tutorial yang menguraikan langkah-langkah pengaturan ABAC, lihat Menggunakan kontrol akses berbasis atribut (ABAC) dalam Panduan Pengguna IAM.

Menggunakan kredensyal sementara dengan Step Functions

Mendukung kredensial sementara	Ya

Beberapa Layanan AWS tidak berfungsi saat Anda masuk menggunakan kredensil sementara. Untuk informasi tambahan, termasuk yang Layanan AWS bekerja dengan kredensyal sementara, lihat Layanan AWS yang bekerja dengan IAM di Panduan Pengguna IAM.

Anda menggunakan kredensi sementara jika Anda masuk AWS Management Console menggunakan metode apa pun kecuali nama pengguna dan kata sandi. Misalnya, ketika Anda mengakses AWS menggunakan tautan masuk tunggal (SSO) perusahaan Anda, proses tersebut secara otomatis membuat kredensil sementara. Anda juga akan secara otomatis membuat kredensial sementara ketika Anda masuk ke konsol sebagai seorang pengguna dan kemudian beralih peran. Untuk informasi selengkapnya tentang peralihan peran, lihat Peralihan peran (konsol) dalam Panduan Pengguna IAM.

Anda dapat membuat kredensyal sementara secara manual menggunakan API AWS CLI atau AWS . Anda kemudian dapat menggunakan kredensil sementara tersebut untuk mengakses. AWS AWS merekomendasikan agar Anda secara dinamis menghasilkan kredensi sementara alih-alih menggunakan kunci akses jangka panjang. Untuk informasi selengkapnya, lihat Kredensial keamanan sementara di IAM.

Izin utama lintas layanan untuk Step Functions

Mendukung sesi akses maju (FAS)	Ya

Saat Anda menggunakan pengguna atau peran IAM untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Saat menggunakan beberapa layanan, Anda mungkin melakukan tindakan yang kemudian dilanjutkan oleh tindakan lain pada layanan yang berbeda. FAS menggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain Layanan AWS atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat Sesi akses maju.

Peran layanan untuk Step Functions

Mendukung peran layanan	Ya

Peran layanan adalah peran IAM yang diambil oleh sebuah layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, memodifikasi, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan Pengguna IAM.

> **Warning**
>
> Mengubah izin untuk peran layanan dapat merusak fungsionalitas Step Functions. Edit peran layanan hanya jika Step Functions memberikan panduan untuk melakukannya.

Peran terkait layanan untuk Step Functions

Mendukung peran terkait layanan	Tidak

Peran terkait layanan adalah jenis peran layanan yang ditautkan ke. Layanan AWS Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.

Untuk detail tentang pembuatan atau pengelolaan peran terkait layanan, silakan lihat Layanan AWS yang berfungsi dengan IAM. Temukan sebuah layanan dalam tabel yang memiliki `Yes` di kolom

Peran terkait layanan. Pilih tautan Ya untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Bagaimana AWS Step Functions bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke Step Functions, pelajari fitur IAM apa yang tersedia untuk digunakan dengan Step Functions.

Fitur IAM yang dapat Anda gunakan AWS Step Functions

Fitur IAM	Dukungan Step Functions
Kebijakan berbasis identitas	Ya
Kebijakan berbasis sumber daya	Tidak
Tindakan kebijakan	Ya
Sumber daya kebijakan	Ya
kunci-kunci persyaratan kebijakan (spesifik layanan)	Ya
ACL	Tidak
ABAC (tanda dalam kebijakan)	Parsial
Kredensial sementara	Ya
Izin pengguna utama	Ya
Peran layanan	Ya
Peran terkait layanan	Tidak

Untuk mendapatkan tampilan tingkat tinggi tentang cara Step Functions dan AWS layanan lainnya bekerja dengan sebagian besar fitur IAM, lihat AWS layanan yang bekerja dengan IAM di Panduan Pengguna IAM.

Contoh kebijakan berbasis identitas untuk AWS Step Functions

Secara default, pengguna dan peran tidak memiliki izin untuk membuat atau memodifikasi sumber daya Step Functions. Mereka juga tidak dapat melakukan tugas dengan menggunakan AWS Management Console, AWS Command Line Interface (AWS CLI), atau AWS API. Untuk memberikan izin kepada pengguna untuk melakukan tindakan pada sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat menjalankan peran.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM menggunakan contoh dokumen kebijakan JSON ini, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Untuk detail tentang tindakan dan jenis sumber daya yang ditentukan oleh Step Functions, termasuk format ARN untuk setiap jenis sumber daya, lihat Tindakan, Sumber Daya, dan Kunci Kondisi untuk AWS Step Functions dalam Referensi Otorisasi Layanan.

Topik

- Praktik terbaik kebijakan
- Menggunakan konsol Step Functions
- Izinkan pengguna melihat izin mereka sendiri

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus sumber daya Step Functions di akun Anda. Tindakan ini mengenakan biaya kepada Akun AWS Anda. 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 terkelola AWS atau kebijakan terkelola 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 melakukan

ini dengan menentukan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, juga dikenal sebagai izin dengan hak akses paling rendah. Untuk informasi selengkapnya tentang cara menggunakan pengguna IAM untuk mengajukan izin, lihat Kebijakan dan izin di IAM dalam Panduan Pengguna IAM.

- Menggunakan syarat dalam kebijakan IAM untuk membatasi akses lebih lanjut – Anda dapat menambahkan suatu syarat 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 AWS CloudFormation. Untuk informasi selengkapnya, lihat Elemen kebijakan JSON IAM: Syarat dalam Panduan Pengguna IAM.
- Menggunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda guna memastikan izin yang aman dan berfungsi – 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 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 mewajibkan MFA saat operasi API dipanggil, tambahkan kondisi MFA pada kebijakan Anda. Untuk informasi selengkapnya, lihat Mengonfigurasi akses API yang diproteksi MFA dalam Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik dalam IAM, lihat Praktik terbaik keamanan di IAM dalam Panduan Pengguna IAM.

Menggunakan konsol Step Functions

Untuk mengakses AWS Step Functions konsol, Anda harus memiliki set izin minimum. Izin ini harus memungkinkan Anda untuk membuat daftar dan melihat detail tentang sumber Step Functions di sumber daya Anda Akun AWS. Jika Anda membuat kebijakan berbasis identitas yang lebih ketat daripada izin minimum yang diperlukan, konsol tidak akan berfungsi sebagaimana mestinya untuk entitas (pengguna atau peran) dengan kebijakan tersebut.

Anda tidak perlu mengizinkan izin konsol minimum untuk pengguna yang melakukan panggilan hanya ke AWS CLI atau AWS API. Sebaliknya, izinkan akses hanya ke tindakan yang cocok dengan operasi API yang coba dilakukan.

Untuk memastikan bahwa pengguna dan peran masih dapat menggunakan konsol Step Functions, lampirkan juga Step Functions *ConsoleAccess* atau kebijakan *ReadOnly* AWS terkelola ke entitas. Untuk informasi selengkapnya, lihat Menambahkan izin ke pengguna dalam Panduan Pengguna IAM.

Izinkan pengguna melihat izin mereka sendiri

Contoh ini menunjukkan cara membuat kebijakan yang mengizinkan para 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": "*"
        }
    ]
}
```

Kebijakan berbasis identitas untuk Step Functions

Mendukung kebijakan berbasis identitas	Ya

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol tindakan apa yang dapat dilakukan oleh pengguna dan peran, pada sumber daya mana, dan dalam keadaan apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan tindakan dan sumber daya yang diperbolehkan atau ditolak serta kondisi di mana tindakan diperbolehkan atau ditolak. Anda tidak dapat menentukan pengguna utama dalam kebijakan berbasis identitas karena kebijakan tersebut berlaku untuk pengguna atau peran yang dilampirkan. Untuk mempelajari semua elemen yang dapat Anda gunakan dalam kebijakan JSON, lihat Referensi elemen kebijakan JSON IAM dalam Panduan Pengguna IAM.

Contoh kebijakan berbasis identitas untuk Step Functions

Untuk melihat contoh kebijakan berbasis identitas Step Functions, lihat. Contoh kebijakan berbasis identitas untuk AWS Step Functions

Kebijakan berbasis sumber daya dalam Step Functions

Mendukung kebijakan berbasis sumber daya	Tidak

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan tepercaya 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. Untuk sumber

daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh pengguna utama tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan pengguna utama dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Untuk mengaktifkan akses lintas akun, Anda dapat menentukan seluruh akun atau entitas IAM di akun lain sebagai pengguna utama di kebijakan berbasis sumber daya. Menambahkan pengguna utama lintas akun ke kebijakan berbasis sumber daya hanyalah setengah dari membangun hubungan kepercayaan. Ketika prinsipal dan sumber daya berbeda Akun AWS, administrator IAM di akun tepercaya juga harus memberikan izin entitas utama (pengguna atau peran) untuk mengakses sumber daya. Izin diberikan dengan melampirkan kebijakan berbasis identitas ke entitas. Namun, jika kebijakan berbasis sumber daya memberikan akses kepada pengguna utama dalam akun yang sama, tidak diperlukan kebijakan berbasis identitas tambahan. Untuk informasi selengkapnya, lihat Perbedaan peran IAM dari kebijakan berbasis sumber daya dalam Panduan Pengguna IAM.

AWS kebijakan terkelola untuk AWS Step Functions

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 yang dikelola AWS dalam Panduan Pengguna IAM.

AWS kebijakan terkelola: AWSStepFunctionsConsoleFullAccess

Anda dapat melampirkan kebijakan `AWSStepFunctionsConsoleFullAccess` ke identitas IAM Anda.

Kebijakan ini memberikan izin *`administrator`* yang memungkinkan akses pengguna untuk menggunakan konsol Step Functions. Untuk pengalaman konsol penuh, pengguna mungkin juga memerlukan PassRole izin iam: pada peran IAM lain yang dapat diasumsikan oleh layanan.

AWS kebijakan terkelola: AWSStepFunctionsReadOnlyAccess

Anda dapat melampirkan kebijakan `AWSStepFunctionsReadOnlyAccess` ke identitas IAM Anda.

Kebijakan ini memberikan izin *`hanya-baca`* yang memungkinkan pengguna atau peran untuk membuat daftar dan mendeskripsikan mesin status, aktivitas, eksekusi, aktivitas, tag MapRuns, serta alias dan versi mesin status.

AWS kebijakan terkelola: AWSStepFunctionsFullAccess

Anda dapat melampirkan kebijakan `AWSStepFunctionsFullAccess` ke identitas IAM Anda.

Kebijakan ini memberikan izin *`penuh`* kepada pengguna atau peran untuk menggunakan Step Functions API. Untuk akses penuh, pengguna harus memiliki PassRole izin *`iam:`* pada setidaknya satu peran IAM yang dapat diasumsikan oleh layanan.

Step Functions memperbarui kebijakan AWS terkelola

Lihat detail tentang pembaruan kebijakan AWS terkelola untuk Step Functions sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan umpan RSS di halaman Step FunctionsRiwayat dokumen.

Perubahan	Deskripsi	Tanggal
AWSStepFunctionsReadOnlyAccess — Permbaruan ke kebijakan yang sudah ada	Step Functions menambahkan izin baru untuk memungkinkan daftar dan membaca data yang terkait dengan: Tag (ListTagsForResource), Peta Terdistribusi (ListMapRuns,	April 02, 2024

Perubahan	Deskripsi	Tanggal
	DescribeMapRun), Versi dan Alias (DescribeStateMachineAlias, ListStateMachineAliases, ListStateMachineVersions).	
Step Functions mulai melacak perubahan	Step Functions mulai melacak perubahan untuk kebijakan yang AWS dikelola.	April 02, 2024

Membuat peran IAM untuk mesin negara Anda

AWS Step Functions dapat mengeksekusi kode dan mengakses AWS sumber daya (seperti memanggil AWS Lambda fungsi). Untuk menjaga keamanan, Anda harus memberikan akses Step Functions ke sumber daya tersebut dengan menggunakan IAM role.

Tutorial untuk Step FunctionsDalam panduan ini memungkinkan Anda untuk memanfaatkan peran IAM yang dihasilkan secara otomatis yang valid untuk AWS Wilayah tempat Anda membuat mesin status. Namun, Anda dapat membuat peran IAM Anda sendiri untuk mesin negara.

Saat membuat kebijakan IAM untuk digunakan mesin status Anda, kebijakan tersebut harus menyertakan izin yang Anda ingin diasumsikan oleh mesin status. Anda dapat menggunakan kebijakan AWS terkelola yang ada sebagai contoh atau Anda dapat membuat kebijakan khusus dari awal yang memenuhi kebutuhan spesifik Anda. Untuk informasi selengkapnya, lihat Membuat kebijakan IAM di Panduan Pengguna IAM

Untuk membuat IAM role Anda sendiri untuk mesin status, ikuti langkah-langkah di bagian ini.

Dalam contoh ini, Anda membuat IAM role dengan izin untuk memanggil fungsi Lambda.

Buat peran untuk Step Functions

1. Masuk ke Konsol IAM, lalu pilih Peran, Buat peran.
2. Pada halaman Pilih entitas tepercaya, di bawah AWS layanan, pilih Step Functions dari daftar, lalu pilih Berikutnya: Izin.
3. Di halaman Kebijakan izin terlampir, pilih Berikutnya: Tinjau .

4. Pada halaman Tinjau, masukkan `StepFunctionsLambdaRole` untuk Nama Peran, lalu pilih Buat peran.

   IAM role muncul di daftar peran.

Untuk informasi selengkapnya tentang izin dan kebijakan IAM, lihat Manajemen Akses dalam Panduan Pengguna IAM.

Mencegah masalah wakil lintas layanan yang membingungkan

Masalah deputi yang bingung adalah masalah keamanan di mana entitas yang tidak memiliki izin untuk melakukan tindakan dapat memaksa entitas yang lebih istimewa untuk melakukan tindakan. Pada tahun AWS, peniruan lintas layanan dapat mengakibatkan masalah wakil yang membingungkan. Peniruan identitas lintas layanan dapat terjadi ketika satu layanan (layanan yang dipanggil) memanggil layanan lain (layanan yang dipanggil). Jenis peniruan identitas ini dapat terjadi lintas akun dan lintas layanan. Layanan pemanggilan dapat dimanipulasi menggunakan izinnya untuk bertindak pada sumber daya pelanggan lain dengan cara yang seharusnya tidak dilakukannya kecuali bila memiliki izin untuk mengakses.

Untuk mencegah kebingungan deputi, AWS sediakan alat yang membantu Anda melindungi data Anda untuk semua layanan dengan prinsip layanan yang telah diberikan akses ke sumber daya di akun Anda. Bagian ini berfokus pada pencegahan wakil bingung lintas layanan khusus untuk AWS Step Functions; Namun, Anda dapat mempelajari lebih lanjut tentang topik ini di bagian masalah wakil yang membingungkan dari Panduan Pengguna IAM.

Sebaiknya gunakan kunci konteks kondisi `aws:SourceAccount`global `aws:SourceArn`dan global dalam kebijakan sumber daya untuk membatasi izin yang Step Functions memberikan layanan lain untuk mengakses sumber daya Anda. Gunakan `aws:SourceArn` jika Anda hanya ingin satu sumber daya dikaitkan dengan akses lintas layanan. Gunakan `aws:SourceAccount` jika Anda ingin mengizinkan sumber daya apa pun di akun tersebut dikaitkan dengan penggunaan lintas layanan.

Cara paling efektif untuk melindungi dari masalah confused deputy adalah dengan menggunakan kunci konteks kondisi global `aws:SourceArn` dengan ARN lengkap sumber daya. Jika Anda tidak mengetahui ARN lengkap sumber daya, atau jika Anda menentukan beberapa sumber daya, gunakan kunci kondisi konteks `aws:SourceArn` global dengan karakter wildcard (*) untuk bagian ARN yang tidak diketahui. Misalnya, `arn:aws:states:*:111122223333:*`.

Berikut adalah contoh kebijakan tepercaya yang menunjukkan bagaimana Anda dapat menggunakan `aws:SourceArn` dan `aws:SourceAccount` dengan Step Functions untuk mencegah masalah deputi yang membingungkan.

```
{
  "Version":"2012-10-17",
  "Statement":[
     {
        "Effect":"Allow",
        "Principal":{
           "Service":[
              "states.amazonaws.com"
           ]
        },
        "Action":"sts:AssumeRole",
        "Condition":{
           "ArnLike":{
              "aws:SourceArn":"arn:aws:states:us-east-1:111122223333:stateMachine:*"
           },
           "StringEquals":{
              "aws:SourceAccount":"111122223333"
           }
        }
     }
  ]
}
```

Melampirkan Kebijakan Sebaris

Step Functions dapat mengontrol layanan lain secara langsung dalam suatu `Task` keadaan. Lampirkan kebijakan sebaris untuk mengizinkan Step Functions mengakses tindakan API dari layanan yang perlu Anda kendalikan.

1. Buka Konsol IAM, pilih Peran, cari peran Step Functions, lalu pilih peran tersebut.
2. Pilih Tambahkan kebijakan sebaris.
3. Gunakan Editor visual atau tab JSON untuk membuat kebijakan untuk peran Anda.

Untuk informasi selengkapnya tentang AWS Step Functions cara mengontrol AWS layanan lain, lihatMenggunakan AWS Step Functions dengan layanan lain.

> **Note**
>
> Untuk contoh kebijakan IAM yang dibuat oleh konsol Step Functions, lihat Kebijakan IAM untuk layanan terintegrasi.

Membuat Izin IAM Terperinci untuk Pengguna Non-Admin

Kebijakan terkelola default di IAM, sepertiReadOnly, tidak sepenuhnya mencakup semua jenis AWS Step Functions izin. Bagian ini menjelaskan berbagai tipe izin dan menyediakan beberapa konfigurasi contoh.

Step Functions memiliki empat kategori izin. Tergantung pada akses yang ingin diberikan kepada pengguna, Anda dapat mengontrol akses dengan menggunakan izin dalam kategori ini.

Izin Tingkat Layanan

Terapkan ke komponen API yang tidak bertindak pada sumber daya tertentu.

Izin Tingkat Mesin Status

Terapkan ke semua komponen API yang bertindak pada mesin status tertentu.

Izin Tingkat Eksekusi

Terapkan ke semua komponen API yang bertindak pada eksekusi tertentu.

Izin Tingkat Aktivitas

Terapkan ke semua komponen API yang bertindak pada aktivitas tertentu atau pada suatu instans aktivitas tertentu.

Izin Tingkat Layanan

Tingkat izin ini berlaku untuk semua tindakan API yang tidak bertindak pada sumber daya tertentu. Ini termasuk `CreateStateMachine`, `CreateActivity`, `ListStateMachines`, dan `ListActivities`.

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

```
      "Effect": "Allow",
      "Action": [
        "states:ListStateMachines",
        "states:ListActivities",
        "states:CreateStateMachine",
        "states:CreateActivity"
      ],
      "Resource": [
        "arn:aws:states:*:*:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam:::role/my-execution-role"
      ]
    }
  ]
}
```

Izin Tingkat Mesin Status

Tingkat izin ini berlaku untuk semua tindakan API yang bertindak pada mesin status tertentu. Operasi API ini memerlukan Amazon Resource Name (ARN) mesin status sebagai bagian dari permintaan, seperti `DeleteStateMachine`, `DescribeStateMachine`, `StartExecution`, dan `ListExecutions`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeStateMachine",
        "states:StartExecution",
        "states:DeleteStateMachine",
        "states:ListExecutions",
        "states:UpdateStateMachine",
        "states:TestState",
        "states:RevealSecrets"
```

```
      ],
      "Resource": [
        "arn:aws:states:*:*:stateMachine:StateMachinePrefix*"
      ]
    }
  ]
}
```

Izin Tingkat Eksekusi

Tingkat izin ini berlaku untuk semua tindakan API yang bertindak pada eksekusi tertentu. Operasi API ini memerlukan ARN eksekusi sebagai bagian dari permintaan, seperti `DescribeExecution`, `GetExecutionHistory`, dan `StopExecution`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeExecution",
        "states:DescribeStateMachineForExecution",
        "states:GetExecutionHistory",
        "states:StopExecution"
      ],
      "Resource": [
        "arn:aws:states:*:*:execution:*:ExecutionPrefix*"
      ]
    }
  ]
}
```

Izin Tingkat Aktivitas

Tingkat izin ini berlaku untuk semua tindakan API yang bertindak pada aktivitas tertentu atau pada instans tertentu. Operasi API ini memerlukan ARN aktivitas atau token instance sebagai bagian dari permintaan, seperti,, `DeleteActivity` `DescribeActivityGetActivityTask`, dan. `SendTaskHeartbeat`

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

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeActivity",
        "states:DeleteActivity",
        "states:GetActivityTask",
        "states:SendTaskHeartbeat"
      ],
      "Resource": [
        "arn:aws:states:*:*:activity:ActivityPrefix*"
      ]
    }
  ]
}
```

Mengakses sumber daya di tempat lain Akun AWS di alur kerja Anda

Step Functions menyediakan akses lintas akun ke sumber daya yang dikonfigurasi Akun AWS di berbagai alur kerja Anda. Dengan menggunakan integrasi layanan Step Functions, Anda dapat memanggil AWS sumber daya lintas akun apa pun meskipun Layanan AWS tidak mendukung kebijakan berbasis sumber daya atau panggilan lintas akun.

Misalnya, asumsikan Anda memiliki dua Akun AWS, yang disebut Pengembangan dan Pengujian, dalam hal yang sama Wilayah AWS. Menggunakan akses lintas akun, alur kerja Anda di akun Pengembangan dapat mengakses sumber daya, seperti bucket Amazon S3, tabel Amazon DynamoDB, dan fungsi Lambda yang tersedia di akun Pengujian.

> **Important**
>
> Peran IAM dan kebijakan berbasis sumber daya mendelegasikan akses ke seluruh akun hanya dalam satu partisi tunggal. Misalnya, anggap Anda memiliki akun di AS Barat (N. California) dalam partisi `aws` standar. Anda juga memiliki akun di Tiongkok (Beijing) dalam partisi `aws-cn`. Anda tidak dapat menggunakan kebijakan berbasis sumber daya Amazon S3 di akun Anda di Tiongkok (Beijing) untuk memungkinkan akses bagi pengguna dalam akun `aws` standar Anda.

Untuk informasi selengkapnya tentang akses lintas akun, lihat Logika evaluasi kebijakan lintas akun di Panduan Pengguna IAM.

Meskipun masing-masing Akun AWS mempertahankan kontrol penuh atas sumber dayanya sendiri, dengan Step Functions, Anda dapat mengatur ulang, menukar, menambah, atau menghapus langkah-langkah dalam alur kerja Anda tanpa perlu menyesuaikan kode apa pun. Anda dapat melakukan ini bahkan saat proses berubah atau aplikasi berkembang.

Anda juga dapat menjalankan eksekusi mesin status bersarang sehingga tersedia di berbagai akun. Melakukannya secara efisien memisahkan dan mengisolasi alur kerja Anda. Saat Anda menggunakan pola integrasi `.sync`layanan dalam alur kerja yang mengakses alur kerja Step Functions lain di akun lain, Step Functions menggunakan polling yang menggunakan kuota yang ditetapkan. Untuk informasi selengkapnya, lihat Jalankan Tugas (.sync).

> **Note**
>
> Saat ini, integrasi AWS SDK lintas wilayah dan akses AWS sumber daya lintas wilayah tidak tersedia di Step Functions.

Daftar Isi

- Konsep kunci dalam topik ini
- Memanggil sumber daya lintas akun
- Tutorial: Mengakses sumber daya lintas akun AWS
- Akses lintas akun untuk pola integrasi.sync

Konsep kunci dalam topik ini

Peran eksekusi

Peran IAM yang digunakan Step Functions untuk menjalankan kode dan mengakses AWS sumber daya, seperti tindakan Invoke AWS Lambda fungsi.

Integrasi layanan

Tindakan API integrasi AWS SDK yang dapat dipanggil dari dalam `Task` status dalam alur kerja Anda.

akun sumber

An Akun AWS yang memiliki mesin negara dan telah memulai pelaksanaannya.

akun target

Sebuah Akun AWS tempat Anda melakukan panggilan lintas akun.

peran target

Peran IAM dalam akun target yang diasumsikan oleh mesin negara untuk melakukan panggilan ke sumber daya yang dimiliki akun target.

Jalankan Job (.sync)

Pola integrasi layanan yang digunakan untuk memanggil layanan, seperti AWS Batch. Itu juga membuat mesin status Step Functions menunggu pekerjaan selesai sebelum melanjutkan ke status berikutnya. Untuk menunjukkan bahwa Step Functions harus menunggu, tambahkan `.sync` akhiran di `Resource` bidang dalam definisi `Task` status Anda.

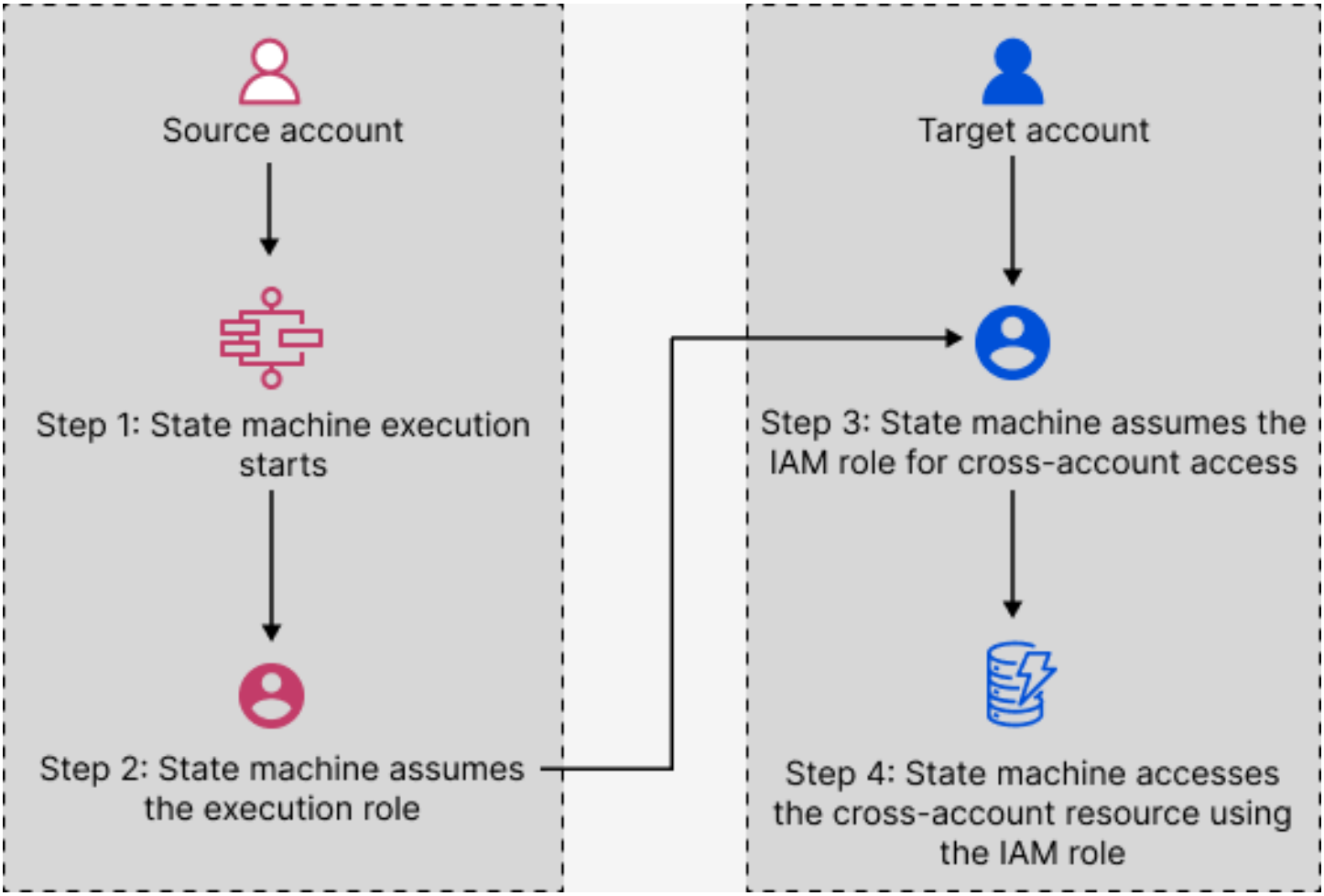
Memanggil sumber daya lintas akun

Untuk memanggil sumber daya lintas akun di alur kerja Anda, lakukan hal berikut:

1. Buat peran IAM di akun target yang berisi sumber daya. Peran ini memberikan akun sumber, yang berisi mesin status, izin untuk mengakses sumber daya akun target.
2. Dalam definisi `Task` status, tentukan peran IAM target yang akan diasumsikan oleh mesin status sebelum menjalankan sumber daya lintas akun.
3. Ubah kebijakan kepercayaan dalam peran IAM target untuk memungkinkan akun sumber mengambil peran ini sementara. Kebijakan kepercayaan harus menyertakan Nama Sumber Daya Amazon (ARN) dari mesin status yang ditentukan dalam akun sumber. Juga, tentukan izin yang sesuai dalam peran IAM target untuk memanggil sumber daya. AWS
4. Perbarui peran eksekusi akun sumber untuk menyertakan izin yang diperlukan untuk mengasumsikan peran IAM target.

Sebagai contoh, lihat Tutorial: Mengakses sumber daya lintas akun AWS.

> **Note**
>
> Anda dapat mengonfigurasi mesin status Anda untuk mengambil peran IAM untuk mengakses sumber daya dari beberapa sumber daya. Akun AWS Namun, mesin negara hanya dapat mengasumsikan satu peran IAM pada waktu tertentu.



Tutorial: Mengakses sumber daya lintas akun AWS

Dengan dukungan akses lintas akun di Step Functions, Anda dapat berbagi sumber daya yang dikonfigurasi secara berbeda Akun AWS. Dalam tutorial ini, kami memandu Anda melalui proses mengakses fungsi Lambda lintas akun yang didefinisikan dalam akun yang disebut Produksi. Fungsi ini dipanggil dari mesin negara dalam akun yang disebut Pengembangan. Dalam tutorial ini, akun Pengembangan disebut sebagai akun sumber dan akun Produksi adalah akun target yang berisi peran IAM target.

Untuk memulai, dalam definisi `Task` status Anda, Anda menentukan peran IAM target yang harus diasumsikan oleh mesin status sebelum menjalankan fungsi Lambda lintas akun. Kemudian, ubah kebijakan kepercayaan dalam peran IAM target untuk memungkinkan akun sumber untuk mengambil peran target sementara. Juga, untuk memanggil AWS sumber daya, tentukan izin yang sesuai dalam peran IAM target. Terakhir, perbarui peran eksekusi akun sumber untuk menentukan izin yang diperlukan untuk mengambil peran target.

Anda dapat mengonfigurasi mesin status Anda untuk mengambil peran IAM untuk mengakses sumber daya dari beberapa sumber daya. Akun AWS Namun, mesin negara hanya dapat mengasumsikan satu peran IAM pada waktu tertentu berdasarkan definisi `Task` negara.

> **Note**
>
> Saat ini, integrasi AWS SDK lintas wilayah dan akses AWS sumber daya lintas wilayah tidak tersedia di Step Functions.

Daftar Isi

- Prasyarat
- Langkah 1: Perbarui definisi status Tugas untuk menentukan peran target
- Langkah 2: Perbarui kebijakan kepercayaan peran target
- Langkah 3: Tambahkan izin yang diperlukan dalam peran target
- Langkah 4: Tambahkan izin dalam peran eksekusi untuk mengambil peran target

Prasyarat

- Tutorial ini menggunakan contoh fungsi Lambda untuk menunjukkan cara mengatur akses lintas akun. Anda dapat menggunakan AWS sumber daya lain, tetapi pastikan Anda telah mengonfigurasi sumber daya di akun yang berbeda.

  > **Important**
  >
  > Peran IAM dan kebijakan berbasis sumber daya mendelegasikan akses ke seluruh akun hanya dalam satu partisi tunggal. Misalnya, anggap Anda memiliki akun di AS Barat (N. California) dalam partisi `aws` standar. Anda juga memiliki akun di Tiongkok (Beijing) dalam partisi `aws-cn`. Anda tidak dapat menggunakan kebijakan berbasis sumber daya Amazon S3 di akun Anda di Tiongkok (Beijing) untuk memungkinkan akses bagi pengguna dalam akun `aws` standar Anda.

- Catat Nama Sumber Daya Amazon (ARN) sumber daya lintas akun dalam file teks. Kemudian dalam tutorial ini, Anda akan memberikan ARN ini dalam definisi status mesin `Task` negara Anda. Berikut ini adalah contoh dari fungsi Lambda ARN:

```
arn:aws:lambda:us-east-2:123456789012:function:functionName
```

- Pastikan Anda telah membuat peran IAM target yang perlu diasumsikan oleh mesin status.

Langkah 1: Perbarui definisi status Tugas untuk menentukan peran target

Dalam `Task` keadaan alur kerja Anda, tambahkan `Credentials` bidang yang berisi identitas yang harus diasumsikan oleh mesin status sebelum menjalankan fungsi Lambda lintas akun.

Prosedur berikut menunjukkan cara mengakses fungsi Lambda lintas akun yang disebut. `Echo` Anda dapat memanggil AWS sumber daya apa pun dengan mengikuti langkah-langkah ini.

1. Buka Konsol Step Functions dan pilih Buat mesin status.
2. Pada halaman Pilih metode authoring, pilih Desain alur kerja Anda secara visual dan simpan semua pilihan default.
3. Untuk membuka Workflow Studio, pilih Berikutnya.
4. Pada tab Tindakan, seret dan lepas `Task` status di kanvas. Ini memanggil fungsi Lambda lintas akun yang menggunakan status ini. `Task`
5. Pada tab Konfigurasi, lakukan hal berikut:

   a. Ubah nama negara menjadi**`Cross-account call`**.

   b. Untuk nama Fungsi, pilih Masukkan nama fungsi, lalu masukkan ARN fungsi Lambda di dalam kotak. Misalnya, `arn:aws:lambda:us-east-2:111122223333:function:Echo`.

   c. Untuk Menyediakan ARN peran IAM, tentukan ARN peran IAM target. Misalnya, `arn:aws:iam::111122223333:role/LambdaRole`.

   > **Tip**
   >
   > Atau, Anda juga dapat menentukan jalur referensi ke pasangan kunci-nilai yang ada di input JSON status yang berisi ARN peran IAM. Untuk melakukan ini, pilih Dapatkan peran IAM ARN saat runtime dari input status. Untuk contoh menentukan nilai dengan menggunakan jalur referensi, lihatMenentukan JSONPath sebagai peran IAM ARN.

6. Pilih Berikutnya.

7. Pada halaman Tinjau kode yang dihasilkan, pilih Berikutnya.
8. Pada halaman Tentukan pengaturan mesin status, tentukan detail untuk mesin status baru, seperti nama, izin, dan tingkat pencatatan.
9. Pilih Buat mesin status.
10. Catat peran IAM mesin negara ARN dan ARN mesin status dalam file teks. Anda harus memberikan ARN ini dalam kebijakan kepercayaan akun target.

Definisi `Task` status Anda sekarang harus terlihat mirip dengan definisi berikut.

```
{
  "StartAt": "Cross-account call",
  "States": {
    "Cross-account call": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "Credentials": {
        "RoleArn": "arn:aws:iam::111122223333:role/LambdaRole"
      },
      "Parameters": {
        "FunctionName": "arn:aws:lambda:us-east-2:111122223333:function:Echo",
      },
      "End": true
    }
  }
}
```

Langkah 2: Perbarui kebijakan kepercayaan peran target

Peran IAM harus ada di akun target dan Anda harus mengubah kebijakan kepercayaannya agar akun sumber dapat mengambil peran ini sementara. Selain itu, Anda dapat mengontrol siapa yang dapat mengambil peran IAM target.

Setelah Anda membuat hubungan kepercayaan, pengguna dari akun sumber dapat menggunakan operasi AssumeRoleAPI AWS Security Token Service (AWS STS). Operasi ini menyediakan kredensil keamanan sementara yang memungkinkan akses ke AWS sumber daya di akun target.

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi konsol, pilih Peran dan kemudian gunakan kotak Pencarian untuk mencari peran IAM target. Misalnya, *LambdaRole*.

3. Pilih tab Trust relationship.
4. Pilih Edit kebijakan kepercayaan dan tempel kebijakan kepercayaan berikut. Pastikan untuk mengganti Akun AWS nomor dan peran IAM ARN. `sts:ExternalId`Bidang lebih lanjut mengontrol siapa yang dapat mengambil peran. Nama mesin status harus menyertakan hanya karakter yang didukung AWS Security Token Service `AssumeRole` API. Untuk informasi selengkapnya, lihat AssumeRoledi Referensi AWS Security Token Service API.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/ExecutionRole"  // The source
 account's state machine execution role ARN
      },
      "Condition": {  // Control which account and state machine can assume the
 target IAM role

        "StringEquals": {
          "sts:ExternalId": "arn:aws:states:us-
east-1:123456789012:stateMachine:testCrossAccount"   //// ARN of the state machine
 that will assume the role.
        }
      }
    }
  ]
}
```

5. Biarkan jendela ini terbuka dan lanjutkan ke langkah berikutnya untuk tindakan lebih lanjut.

Langkah 3: Tambahkan izin yang diperlukan dalam peran target

Izin dalam kebijakan IAM menentukan apakah permintaan tertentu diizinkan atau ditolak. Peran IAM target harus memiliki izin yang benar untuk menjalankan fungsi Lambda.

1. Pilih tab Izin.
2. Pilih Tambahkan izin, lalu pilih Buat kebijakan sebaris.

3. Pilih tab JSON dan ganti konten yang ada dengan izin berikut. Pastikan untuk mengganti ARN fungsi Lambda Anda.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:us-east-2:111122223333:function:Echo"  // The
 cross-account AWS resource being accessed
    }
  ]
}
```

4. Pilih Tinjau kebijakan.
5. Pada halaman Kebijakan ulasan, masukkan nama untuk izin, lalu pilih Buat kebijakan.

Langkah 4: Tambahkan izin dalam peran eksekusi untuk mengambil peran target

Step Functions tidak secara otomatis menghasilkan AssumeRolekebijakan untuk semua integrasi layanan lintas akun. Anda harus menambahkan izin yang diperlukan dalam peran eksekusi mesin status untuk memungkinkannya mengambil peran IAM target dalam satu atau lebih Akun AWS.

1. Buka peran eksekusi mesin status Anda di konsol IAM di https://console.aws.amazon.com/iam/. Untuk melakukannya:
   a. Buka mesin status yang Anda buat di Langkah 1 di akun sumber.
   b. Pada halaman detail mesin Negara, pilih peran IAM ARN.
2. Pada tab Izin, pilih Tambahkan izin, lalu pilih Buat kebijakan sebaris.
3. Pilih tab JSON dan ganti konten yang ada dengan izin berikut. Pastikan untuk mengganti ARN fungsi Lambda Anda.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
```

```
      "Resource": "arn:aws:iam::111122223333:role/LambdaRole"  // The target role
 to be assumed
    }
  ]
}
```

4. Pilih Tinjau kebijakan.
5. Pada halaman Kebijakan ulasan, masukkan nama untuk izin, lalu pilih Buat kebijakan.

Akses lintas akun untuk pola integrasi.sync

Saat Anda menggunakan pola integrasi `.sync` layanan dalam alur kerja, Step Functions akan melakukan polling sumber daya lintas akun yang dipanggil untuk mengonfirmasi bahwa tugas telah selesai. Hal ini menyebabkan sedikit keterlambatan antara waktu penyelesaian tugas yang sebenarnya dan waktu ketika Step Functions mengenali tugas sebagai selesai. Peran IAM target memerlukan izin yang diperlukan untuk `.sync` pemanggilan untuk menyelesaikan polling loop ini. Untuk melakukan ini, peran IAM target harus memiliki kebijakan kepercayaan yang memungkinkan akun sumber untuk mengasumsikannya. Selain itu, peran IAM target memerlukan izin yang diperlukan untuk menyelesaikan polling loop.

> **Note**
>
> Untuk Alur Kerja Ekspres bersarang, saat ini `arn:aws:states:::states:startExecution.sync` tidak didukung. Gunakan `arn:aws:states:::aws-sdk:sfn:startSyncExecution` sebagai gantinya.

Pembaruan kebijakan kepercayaan untuk panggilan.sync

Perbarui kebijakan kepercayaan peran IAM target Anda seperti yang ditunjukkan pada contoh berikut. `sts:ExternalId`Bidang lebih lanjut mengontrol siapa yang dapat mengambil peran. Nama mesin status harus menyertakan hanya karakter yang didukung AWS Security Token Service `AssumeRole` API. Untuk informasi selengkapnya, lihat AssumeRoledi Referensi AWS Security Token Service API.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": "sts:AssumeRole",
      "Principal": {
        "AWS": "arn:aws:iam::sourceAccountID:role/InvokeRole",
      },
      "Condition": {
        "StringEquals": {
          "sts:ExternalId": "arn:aws:states:us-
east-2:sourceAccountID:stateMachine:stateMachineName"
        }
      }
    }
  ]
}
```

Izin diperlukan untuk panggilan.sync

Untuk memberikan izin yang diperlukan untuk mesin status Anda, perbarui izin yang diperlukan untuk peran IAM target. Untuk informasi selengkapnya, lihat the section called “Kebijakan IAM untuk layanan terintegrasi”. EventBridge Izin Amazon dari kebijakan contoh tidak diperlukan. Misalnya, untuk memulai mesin status, tambahkan izin berikut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution"
      ],
      "Resource": [
        "arn:aws:states:region:accountID:stateMachine:stateMachineName"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeExecution",
        "states:StopExecution"
      ],
      "Resource": [
        "arn:aws:states:region:accountID:execution:stateMachineName:*"
      ]
    }
```

```
  ]
}
```

Amazon VPC Endpoints untuk Step Functions

Jika Anda menggunakan Amazon Virtual Private Cloud (Amazon VPC) untuk meng-host AWS sumber daya Anda, Anda dapat membuat koneksi antara Amazon VPC dan alur kerja. AWS Step Functions Anda dapat menggunakan koneksi ini dengan alur kerja Step Functions tanpa menyeberangi internet publik. Titik akhir Amazon VPC didukung oleh Alur Kerja Standar, Alur Kerja Ekspres, dan Alur Kerja Ekspres Sinkron.

Amazon VPC memungkinkan Anda meluncurkan AWS sumber daya di jaringan virtual khusus. Anda dapat menggunakan VPC untuk mengendalikan pengaturan jaringan, seperti rentang alamat IP, subnet, tabel rute, dan gateway jaringan. Untuk informasi lebih lanjut tentang Amazon VPC, lihat Panduan Pengguna Amazon VPC.

Untuk menghubungkan Amazon VPC Anda ke Step Functions, Anda harus terlebih dahulu menentukan titik akhir VPC antarmuka, yang memungkinkan Anda menghubungkan VPC Anda ke layanan lain. AWS Titik akhir memberikan konektivitas yang dapat andal, dapat diskalakan, tanpa memerlukan gateway internet, instans terjemahan alamat jaringan (NAT), atau koneksi VPN. Untuk informasi selengkapnya, lihat VPC Endpoint Antarmuka (AWS PrivateLink) dalam Panduan Pengguna Amazon VPC.

Membuat Titik Akhir

Anda dapat membuat AWS Step Functions endpoint di VPC menggunakan, AWS Management Console AWS Command Line Interface the AWS CLI(), SDK, AWS API, AWS Step Functions atau. AWS CloudFormation

Untuk informasi tentang membuat dan mengonfigurasi titik akhir menggunakan konsol Amazon VPC atau AWS CLI, lihat Membuat Titik Akhir Antarmuka dalam Panduan Pengguna Amazon VPC.

> **Note**
>
> Ketika Anda membuat titik akhir, tetapkan Step Functions sebagai layanan yang ingin Anda hubungkan ke VPC Anda. Di konsol VPC Amazon, nama layanan bervariasi berdasarkan Wilayah. AWS Misalnya, jika Anda memilih US East (Virginia N.), nama layanan untuk Alur Kerja Standar dan Alur Kerja Ekspres adalah com.amazonaws.us-east-1.states, dan nama layanan untuk Alur Kerja Ekspres Sinkron adalah com.amazonaws.us-east-1.sync-states.

> **Note**
>
> Dimungkinkan untuk menggunakan Titik Akhir VPC tanpa mengganti titik akhir di SDK melalui DNS Pribadi. Namun, jika Anda ingin mengganti titik akhir di SDK untuk Alur Kerja Synchronous Express, Anda perlu mengatur konfigurasi ke. `DisableHostPrefixInjection` `true` Contoh (Java SDK V2):
>
> ```
> SfnClient.builder()
>   .endpointOverride(URI.create("https://vpce-{vpceId}.sync-states.us-
> east-1.vpce.amazonaws.com"))
>   .overrideConfiguration(ClientOverrideConfiguration.builder()
>
>  .advancedOptions(ImmutableMap.of(SdkAdvancedClientOption.DISABLE_HOST_PREFIX_INJECTION,
>  true))
>     .build())
>   .build();
> ```

Untuk informasi tentang membuat dan mengonfigurasi titik akhir menggunakan AWS CloudFormation, lihat sumber daya AWS: :EC2: :vpcendPoint di Panduan Pengguna.AWS CloudFormation

Kebijakan Amazon VPC Endpoint

Untuk mengontrol akses konektivitas ke Step Functions, Anda dapat melampirkan kebijakan endpoint AWS Identity and Access Management (IAM) saat membuat endpoint Amazon VPC. Anda dapat membuat aturan IAM kompleks dengan melampirkan beberapa kebijakan titik akhir. Untuk informasi selengkapnya, lihat:

- Kebijakan Amazon Virtual Private Cloud Endpoint untuk Step Functions
- Membuat Izin IAM Terperinci untuk Pengguna Non-Admin
- Mengontrol Akses ke Layanan dengan Titik Akhir VPC

Kebijakan Amazon Virtual Private Cloud Endpoint untuk Step Functions

Anda dapat membuat kebijakan endpoint Amazon VPC untuk Step Functions yang Anda tentukan sebagai berikut:

- Prinsipal yang dapat melakukan tindakan.

- Tindakan yang dapat dilakukan.
- Sumber daya untuk melakukan tindakan.

Contoh berikut menunjukkan kebijakan titik akhir VPC Amazon yang memungkinkan satu pengguna membuat mesin status, dan menolak semua izin pengguna lain untuk menghapus mesin status. Kebijakan contoh juga memberikan izin eksekusi kepada semua pengguna.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "*Execution",
            "Resource": "*",
            "Effect": "Allow",
            "Principal": "*"
        },
        {
            "Action": "states:CreateStateMachine",
            "Resource": "*",
            "Effect": "Allow",
            "Principal": {
              "AWS": "arn:aws:iam::123456789012:user/MyUser"
            }
        },
        {
            "Action": "states:DeleteStateMachine",
            "Resource": "*",
            "Effect": "Deny",
            "Principal": "*"
        }
    ]
}
```

Untuk informasi selengkapnya tentang cara membuat kebijakan titik akhir, lihat hal berikut:

- Membuat Izin IAM Terperinci untuk Pengguna Non-Admin
- Mengontrol Akses ke Layanan dengan Titik Akhir VPC

Kebijakan IAM untuk layanan terintegrasi

Saat Anda membuat mesin status di AWS Step Functions konsol, Step Functions menghasilkan kebijakan AWS Identity and Access Management (IAM) berdasarkan sumber daya yang digunakan dalam definisi mesin status Anda sebagai berikut:

- Jika mesin status Anda menggunakan salah satu integrasi yang Dioptimalkan, Step Functions akan membuat kebijakan dengan izin dan peran yang diperlukan untuk mesin status Anda. (Pengecualian: MediaConvert integrasi mengharuskan Anda mengatur izin secara manual — lihatKebijakan IAM untuk AWS Elemental MediaConvert.)
- Jika mesin status Anda menggunakan salah satu integrasi AWS SDK, peran IAM dengan izin sebagian akan dibuat. Setelah itu, Anda dapat menggunakan konsol IAM untuk menambahkan kebijakan peran yang hilang.

Contoh berikut menunjukkan cara Step Functions menghasilkan kebijakan IAM berdasarkan ketetapan mesin status Anda. Item dalam kode contoh seperti *`[[resourceName]]`* diganti dengan sumber daya statis yang tercantum dalam ketetapan mesin status Anda. Jika Anda memiliki beberapa sumber daya statis, akan ada entri untuk masing-masing dalam IAM role.

Sumber Daya Dinamis vs. Statis

Sumber daya statis ditetapkan secara langsung dalam status tugas dari mesin status Anda. Bila Anda menyertakan informasi tentang sumber daya yang ingin Anda panggil langsung dalam status tugas Anda, Step Functions membuat peran IAM hanya untuk sumber daya tersebut.

Sumber daya dinamis adalah sumber daya yang dilewatkan ke input status Anda, dan diakses menggunakan Jalur (lihat Jalan). Jika Anda melewati sumber daya dinamis untuk tugas Anda, Step Functions akan membuat kebijakan yang lebih istimewa yang menentukan: `"Resource": "*"`.

Izin tambahan untuk tugas menggunakan pola Jalankan Tugas

Untuk tugas yang menggunakan pola Jalankan Tugas (yang berakhir di `.sync`), izin tambahan diperlukan untuk memantau dan menerima respons dari tindakan API layanan yang terhubung. Kebijakan terkait mencakup lebih banyak izin daripada untuk tugas-tugas yang menggunakan pola Minta Respons atau Tunggu untuk Panggil Balik. Lihat Pola integrasi layanan untuk informasi tentang tugas-tugas sinkron.

> **Note**
>
> Anda perlu memberikan izin tambahan untuk integrasi layanan yang mendukung pola Run a Job (.sync).

Step Functions menggunakan dua metode untuk memantau status tugas ketika tugas dijalankan pada layanan terhubung, polling dan peristiwa.

Polling memerlukan izin untuk Tindakan API `Describe` atau `Get`, seperti `ecs:DescribeTasks` atau `glue:GetJobRun`. Jika izin ini hilang dari peran Anda, maka Step Functions mungkin tidak dapat menentukan status tugas Anda. Ini karena beberapa integrasi layanan Run a Job (.sync) tidak mendukung EventBridge peristiwa, dan beberapa layanan hanya mengirim acara dengan upaya terbaik.

Peristiwa yang dikirim dari AWS layanan ke Amazon EventBridge diarahkan ke Step Functions menggunakan aturan terkelola, dan memerlukan izin untuk `events:PutTargetsevents:PutRule`, dan`events:DescribeRule`. Jika izin ini hilang dari peran Anda, mungkin ada penundaan sebelum Step Functions menyadari penyelesaian tugas Anda. Untuk informasi selengkapnya tentang EventBridge acara, lihat Acara dari AWS layanan.

> **Note**
>
> Untuk tugas Jalankan Job (.sync) yang mendukung polling dan acara, tugas Anda mungkin masih selesai dengan benar menggunakan peristiwa. Hal ini dapat terjadi bahkan jika peran Anda tidak memiliki izin yang diperlukan untuk polling. Dalam kasus ini, Anda mungkin tidak segera melihat bahwa izin pemungutan suara salah atau hilang. Dalam instans langka saat peristiwa gagal dikirim ke atau diproses oleh Step Functions, eksekusi Anda bisa menjadi macet. Untuk memverifikasi bahwa izin polling dikonfigurasi dengan benar, Anda dapat menjalankan eksekusi di lingkungan tanpa EventBridge peristiwa dengan cara berikut:
>
> - Hapus aturan terkelola dari EventBridge, yang bertanggung jawab untuk meneruskan peristiwa ke Step Functions. Aturan terkelola ini dibagikan oleh semua mesin status di akun Anda, sehingga Anda harus melakukan tindakan ini hanya dalam akun pengujian atau pengembangan untuk menghindari dampak yang tidak diinginkan pada mesin status lainnya. Anda dapat mengidentifikasi aturan terkelola tertentu untuk dihapus dengan memeriksa bidang `Resource` yang digunakan untuk `events:PutRule` pada templat kebijakan untuk layanan target. Aturan terkelola akan dibuat ulang saat Anda membuat

atau memperbarui mesin status berikutnya yang menggunakan integrasi layanan tersebut. Untuk informasi selengkapnya tentang menghapus EventBridge aturan, lihat Menonaktifkan atau menghapus aturan.

- Uji dengan Step Functions Local, yang tidak mendukung penggunaan peristiwa untuk menyelesaikan tugas Run a Job (.sync). Untuk menggunakan Step Functions Lokal, asumsikan IAM role digunakan oleh mesin status Anda. Anda mungkin perlu mengedit Hubungan Kepercayaan. Atur variabel lingkungan `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, dan `AWS_SESSION_TOKEN` ke nilai peran yang diasumsikan ini, kemudian jalankan Step Functions Lokal menggunakan java -jar StepFunctionsLocal.jar. Terakhir, gunakan --endpoint-url parameter AWS CLI with the untuk membuat mesin state, memulai eksekusi, dan mendapatkan riwayat eksekusi. Untuk informasi selengkapnya, lihat Menguji mesin negara secara lokal.

Jika tugas yang menggunakan pola Run a Job (.sync) dihentikan, Step Functions akan melakukan upaya terbaik untuk membatalkan tugas tersebut. Hal ini memerlukan izin untuk tindakan API `Cancel`, `Stop`, `Terminate`, atau `Delete`, seperti `batch:TerminateJob` atau `eks:DeleteCluster`. Jika izin ini hilang dari peran Anda, Step Functions tidak akan dapat membatalkan tugas dan Anda dapat terkena biaya tambahan sementara tugas terus berjalan. Untuk informasi selengkapnya tentang menghentikan tugas, lihat Jalankan Tugas.

Templat kebijakan yang digunakan untuk membuat IAM role

Topik berikut mencakup templat kebijakan yang digunakan ketika Anda memilih agar Step Functions membuat peran baru untuk Anda.

> **Note**
>
> Tinjau template ini untuk memahami cara Step Functions membuat kebijakan IAM Anda, dan sebagai contoh cara membuat kebijakan IAM secara manual untuk Step Functions saat bekerja dengan layanan lain AWS . Untuk informasi selengkapnya tentang integrasi layanan Step Functions, lihat Menggunakan AWS Step Functions dengan layanan lain.

Topik

- Kebijakan IAM untuk Amazon API Gateway
- Kebijakan IAM untuk Amazon Athena

- Kebijakan IAM untuk AWS Batch
- IAM policies for Amazon Bedrock
- Kebijakan IAM untuk AWS CodeBuild
- Kebijakan IAM untuk Amazon DynamoDB
- Kebijakan IAM untuk Amazon ECS/AWS Fargate
- Kebijakan IAM untuk Amazon EKS
- Kebijakan IAM untuk Amazon EMR
- Kebijakan IAM untuk Amazon EMR di EKS
- Kebijakan IAM untuk Amazon EMR Serverless
- Kebijakan IAM untuk Amazon EventBridge
- Kebijakan IAM untuk AWS Lambda
- Kebijakan IAM untuk AWS Elemental MediaConvert
- Kebijakan IAM untuk AWS Glue
- Kebijakan IAM untuk AWS Glue DataBrew
- Kebijakan IAM untuk Amazon SageMaker
- Kebijakan IAM untuk Amazon SNS
- Kebijakan IAM untuk Amazon SQS
- Kebijakan IAM untuk AWS Step Functions
- Kebijakan IAM untuk AWS X-Ray
- Aktivitas atau Tidak Ada Tugas

Kebijakan IAM untuk Amazon API Gateway

Contoh templat berikut menunjukkan cara AWS Step Functions menghasilkan kebijakan IAM berdasarkan sumber daya dalam definisi mesin status Anda. Lihat informasi yang lebih lengkap di Kebijakan IAM untuk layanan terintegrasi dan Pola integrasi layanan.

Sumber Daya:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "execute-api:Invoke"
            ],
            "Resource": [
                "arn:[[region]]:[[accountId]]:*"
            ]
        }
    ]
}
```

Contoh kode berikut ini menunjukkan kebijakan sumber daya untuk memanggil API Gateway.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "states.amazonaws.com"
            },
            "Action": "execute-api:Invoke",
            "Resource": "arn:aws:execute-api:<region>:<account-id>:<api-id>/<stage-
name>/<HTTP-VERB>/<resource-path-specifier>",
            "Condition": {
                "StringEquals": {
                    "aws:SourceArn": [
                        "<SourceStateMachineArn>"
                    ]
                }
            }
        }
    ]
}
```

Kebijakan IAM untuk Amazon Athena

Contoh templat berikut menunjukkan cara AWS Step Functions menghasilkan kebijakan IAM berdasarkan sumber daya dalam definisi mesin status Anda. Lihat informasi yang lebih lengkap di Kebijakan IAM untuk layanan terintegrasi dan Pola integrasi layanan.

`StartQueryExecution`

Sumber daya statis

Run a Job (.sync)

```
{
    "Version": "2012-10-17",
    "Statement":[
    {
        "Effect": "Allow",
        "Action": [
            "athena:startQueryExecution",
            "athena:stopQueryExecution",
            "athena:getQueryExecution",
            "athena:getDataCatalog"
        ],
        "Resource": [
            "arn:aws:athena:{{region}}:{{accountId}}:workgroup/[[workGroup]]",
            "arn:aws:athena:{{region}}:{{accountId}}:datacatalog/*"
        ]
    },
        {
        "Effect": "Allow",
        "Action": [
            "s3:GetBucketLocation",
            "s3:GetObject",
            "s3:ListBucket",
            "s3:ListBucketMultipartUploads",
            "s3:ListMultipartUploadParts",
            "s3:AbortMultipartUpload",
            "s3:CreateBucket",
            "s3:PutObject"
        ],
        "Resource": [
            "arn:aws:s3:::*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "glue:CreateDatabase",
            "glue:GetDatabase",
            "glue:GetDatabases",
            "glue:UpdateDatabase",
            "glue:DeleteDatabase",
            "glue:CreateTable",
            "glue:UpdateTable",
```

```
            "glue:GetTable",
            "glue:GetTables",
            "glue:DeleteTable",
            "glue:BatchDeleteTable",
            "glue:BatchCreatePartition",
            "glue:CreatePartition",
            "glue:UpdatePartition",
            "glue:GetPartition",
            "glue:GetPartitions",
            "glue:BatchGetPartition",
            "glue:DeletePartition",
            "glue:BatchDeletePartition"
        ],
        "Resource": [
            "arn:aws:glue:{{region}}:{{accountId}}:catalog",
            "arn:aws:glue:{{region}}:{{accountId}}:database/*",
            "arn:aws:glue:{{region}}:{{accountId}}:table/*",
            "arn:aws:glue:{{region}}:{{accountId}}:userDefinedFunction/*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "lakeformation:GetDataAccess"
        ],
        "Resource": [
            "*"
        ]
    }
]
}
```

Request Response

```
{
    "Version": "2012-10-17",
    "Statement":[
    {
        "Effect": "Allow",
        "Action": [
            "athena:startQueryExecution",
            "athena:getDataCatalog"
        ],
```

```
        "Resource": [
            "arn:aws:athena:{{region}}:{{accountId}}:workgroup/[[workGroup]]",
            "arn:aws:athena:{{region}}:{{accountId}}:datacatalog/*"
        ]
    },
        {
        "Effect": "Allow",
        "Action": [
            "s3:GetBucketLocation",
            "s3:GetObject",
            "s3:ListBucket",
            "s3:ListBucketMultipartUploads",
            "s3:ListMultipartUploadParts",
            "s3:AbortMultipartUpload",
            "s3:CreateBucket",
            "s3:PutObject"
        ],
        "Resource": [
            "arn:aws:s3:::*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "glue:CreateDatabase",
            "glue:GetDatabase",
            "glue:GetDatabases",
            "glue:UpdateDatabase",
            "glue:DeleteDatabase",
            "glue:CreateTable",
            "glue:UpdateTable",
            "glue:GetTable",
            "glue:GetTables",
            "glue:DeleteTable",
            "glue:BatchDeleteTable",
            "glue:BatchCreatePartition",
            "glue:CreatePartition",
            "glue:UpdatePartition",
            "glue:GetPartition",
            "glue:GetPartitions",
            "glue:BatchGetPartition",
            "glue:DeletePartition",
            "glue:BatchDeletePartition"
        ],
```

```
        "Resource": [
            "arn:aws:glue:{{region}}:{{accountId}}:catalog",
            "arn:aws:glue:{{region}}:{{accountId}}:database/*",
            "arn:aws:glue:{{region}}:{{accountId}}:table/*",
            "arn:aws:glue:{{region}}:{{accountId}}:userDefinedFunction/*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "lakeformation:GetDataAccess"
        ],
        "Resource": [
            "*"
        ]
    }
]
}
```

Sumber daya dinamis

Run a Job (.sync)

```
{
    "Version": "2012-10-17",
    "Statement":[
    {
        "Effect": "Allow",
        "Action": [
            "athena:startQueryExecution",
            "athena:stopQueryExecution",
            "athena:getQueryExecution",
            "athena:getDataCatalog"
        ],
        "Resource": [
            "arn:aws:athena:{{region}}:{{accountId}}:workgroup/*",
            "arn:aws:athena:{{region}}:{{accountId}}:datacatalog/*"
        ]
    },
        {
        "Effect": "Allow",
        "Action": [
```

```
            "s3:GetBucketLocation",
            "s3:GetObject",
            "s3:ListBucket",
            "s3:ListBucketMultipartUploads",
            "s3:ListMultipartUploadParts",
            "s3:AbortMultipartUpload",
            "s3:CreateBucket",
            "s3:PutObject"
        ],
        "Resource": [
            "arn:aws:s3:::*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "glue:CreateDatabase",
            "glue:GetDatabase",
            "glue:GetDatabases",
            "glue:UpdateDatabase",
            "glue:DeleteDatabase",
            "glue:CreateTable",
            "glue:UpdateTable",
            "glue:GetTable",
            "glue:GetTables",
            "glue:DeleteTable",
            "glue:BatchDeleteTable",
            "glue:BatchCreatePartition",
            "glue:CreatePartition",
            "glue:UpdatePartition",
            "glue:GetPartition",
            "glue:GetPartitions",
            "glue:BatchGetPartition",
            "glue:DeletePartition",
            "glue:BatchDeletePartition"
        ],
        "Resource": [
            "arn:aws:glue:{{region}}:{{accountId}}:catalog",
            "arn:aws:glue:{{region}}:{{accountId}}:database/*",
            "arn:aws:glue:{{region}}:{{accountId}}:table/*",
            "arn:aws:glue:{{region}}:{{accountId}}:userDefinedFunction/*"
        ]
    },
    {
```

```
        "Effect": "Allow",
        "Action": [
            "lakeformation:GetDataAccess"
        ],
        "Resource": [
            "*"
        ]
    }
]
}
```

Request Response

```
{
    "Version": "2012-10-17",
    "Statement":[
    {
        "Effect": "Allow",
        "Action": [
            "athena:startQueryExecution",
            "athena:getDataCatalog"
        ],
        "Resource": [
            "arn:aws:athena:{{region}}:{{accountId}}:workgroup/*",
            "arn:aws:athena:{{region}}:{{accountId}}:datacatalog/*"
        ]
    },
        {
        "Effect": "Allow",
        "Action": [
            "s3:GetBucketLocation",
            "s3:GetObject",
            "s3:ListBucket",
            "s3:ListBucketMultipartUploads",
            "s3:ListMultipartUploadParts",
            "s3:AbortMultipartUpload",
            "s3:CreateBucket",
            "s3:PutObject"
        ],
        "Resource": [
            "arn:aws:s3:::*"
        ]
    },
```

```
    {
        "Effect": "Allow",
        "Action": [
            "glue:CreateDatabase",
            "glue:GetDatabase",
            "glue:GetDatabases",
            "glue:UpdateDatabase",
            "glue:DeleteDatabase",
            "glue:CreateTable",
            "glue:UpdateTable",
            "glue:GetTable",
            "glue:GetTables",
            "glue:DeleteTable",
            "glue:BatchDeleteTable",
            "glue:BatchCreatePartition",
            "glue:CreatePartition",
            "glue:UpdatePartition",
            "glue:GetPartition",
            "glue:GetPartitions",
            "glue:BatchGetPartition",
            "glue:DeletePartition",
            "glue:BatchDeletePartition"
        ],
        "Resource": [
            "arn:aws:glue:{{region}}:{{accountId}}:catalog",
            "arn:aws:glue:{{region}}:{{accountId}}:database/*",
            "arn:aws:glue:{{region}}:{{accountId}}:table/*",
            "arn:aws:glue:{{region}}:{{accountId}}:userDefinedFunction/*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "lakeformation:GetDataAccess"
        ],
        "Resource": [
            "*"
        ]
    }
]
}
```

StopQueryExecution

Sumber Daya

```
{
    "Version": "2012-10-17",
    "Statement":[
    {
        "Effect": "Allow",
        "Action": [
            "athena:stopQueryExecution"
        ],
        "Resource": [
            "arn:aws:athena:{{region}}:{{accountId}}:workgroup/*"
        ]
    }
    ]
}
```

GetQueryExecution

Sumber Daya

```
{
    "Version": "2012-10-17",
    "Statement":[
    {
        "Effect": "Allow",
        "Action": [
            "athena:getQueryExecution"
        ],
        "Resource": [
            "arn:aws:athena:{{region}}:{{accountId}}:workgroup/*"
        ]
    }
    ]
}
```

GetQueryResults

Sumber Daya

```
{
```

```
    "Version": "2012-10-17",
    "Statement":[
    {
        "Effect": "Allow",
        "Action": [
            "athena:getQueryResults"
        ],
        "Resource": [
            "arn:aws:athena:{{region}}:{{accountId}}:workgroup/*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:GetObject"
        ],
        "Resource": [
            "arn:aws:s3:::*"
        ]
    }
    ]
}
```

Kebijakan IAM untuk AWS Batch

Contoh templat berikut menunjukkan cara AWS Step Functions menghasilkan kebijakan IAM berdasarkan sumber daya dalam definisi mesin status Anda. Lihat informasi yang lebih lengkap di Kebijakan IAM untuk layanan terintegrasi dan Pola integrasi layanan.

Karena AWS Batch memberikan dukungan sebagian untuk kontrol akses tingkat sumber daya, Anda harus menggunakannya. `"Resource": "*"`

Run a Job (.sync)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "batch:SubmitJob",
                "batch:DescribeJobs",
                "batch:TerminateJob"
```

```
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
               "arn:aws:events:[[region]]:[[accountId]]:rule/
StepFunctionsGetEventsForBatchJobsRule"
            ]
        }
    ]
}
```

Request Response

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "batch:SubmitJob"
            ],
            "Resource": "*"
        }
    ]
}
```

IAM policies for Amazon Bedrock

Saat Anda membuat mesin status menggunakan konsol, Step Functions secara otomatis membuat peran eksekusi untuk mesin status Anda dengan hak istimewa paling sedikit yang diperlukan. IAMPeran yang dihasilkan secara otomatis ini berlaku untuk Wilayah AWS di mana Anda membuat mesin status.

Contoh templat berikut menunjukkan cara AWS Step Functions menghasilkan kebijakan IAM berdasarkan sumber daya dalam definisi mesin status Anda. Lihat informasi yang lebih lengkap di Kebijakan IAM untuk layanan terintegrasi dan Pola integrasi layanan.

Sebaiknya saat Anda membuat IAM kebijakan, jangan sertakan wildcard dalam kebijakan. Sebagai praktik terbaik keamanan, Anda harus mencatat kebijakan Anda sebanyak mungkin. Anda harus menggunakan kebijakan dinamis hanya jika parameter input tertentu tidak diketahui selama runtime.

Dalam topik ini:

- IAMcontoh kebijakan untuk Amazon Bedrock integrasi dengan Step Functions

IAMcontoh kebijakan untuk Amazon Bedrock integrasi dengan Step Functions

Bagian berikut menjelaskan IAM izin yang Anda perlukan berdasarkan Amazon Bedrock API yang Anda gunakan untuk fondasi tertentu atau model yang disediakan. Bagian ini juga berisi contoh kebijakan yang memberikan akses penuh.

Ingatlah untuk mengganti teks yang *dicetak miring dengan informasi* spesifik sumber daya Anda.

- IAMcontoh kebijakan untuk mengakses model pondasi tertentu menggunakan InvokeModel
- IAMcontoh kebijakan untuk mengakses model tertentu yang disediakan menggunakan InvokeModel
- Contoh IAM kebijakan akses penuh untuk digunakan InvokeModel
- IAMcontoh kebijakan untuk mengakses model pondasi tertentu sebagai model dasar
- IAMcontoh kebijakan untuk mengakses model kustom tertentu sebagai model dasar
- Contoh IAM kebijakan akses penuh untuk CreateModelCustomizationJob menggunakan.sync
- IAMcontoh kebijakan untuk mengakses model fondasi tertentu menggunakan CreateModelCustomizationJob .sync
- IAMcontoh kebijakan untuk mengakses model kustom CreateModelCustomizationJob menggunakan.sync
- Contoh IAM kebijakan akses penuh untuk CreateModelCustomizationJob menggunakan.sync

IAMcontoh kebijakan untuk mengakses model pondasi tertentu menggunakan InvokeModel

Berikut ini adalah contoh IAM kebijakan untuk mesin status yang mengakses model dasar tertentu bernama `amazon.titan-text-express-v1` menggunakan tindakan InvokeModelAPI.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Sid": "InvokeModel1",
            "Action": [
                "bedrock:InvokeModel"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2::foundation-model/amazon.titan-text-express-v1"
            ]
        }
    ]
}
```

IAMcontoh kebijakan untuk mengakses model tertentu yang disediakan menggunakan InvokeModel

Berikut ini adalah contoh IAM kebijakan untuk mesin status yang mengakses model tertentu yang disediakan bernama `c2oi931ulksx` menggunakan tindakan API. InvokeModel

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Sid": "InvokeModel1",
      "Action": [
        "bedrock:InvokeModel"
      ],
      "Resource": [
        "arn:aws:bedrock:us-east-2:123456789012:provisioned-model/c2oi931ulksx"
      ]
    }
  ]
}
```

Contoh IAM kebijakan akses penuh untuk digunakan InvokeModel

Berikut ini adalah contoh IAM kebijakan untuk mesin status yang menyediakan akses penuh saat Anda menggunakan tindakan InvokeModelAPI.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Sid": "InvokeModel1",
            "Action": [
                "bedrock:InvokeModel"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2::foundation-model/*",
                "arn:aws:bedrock:us-east-2:123456789012:provisioned-model/*"
            ]
        }
    ]
}
```

IAMcontoh kebijakan untuk mengakses model pondasi tertentu sebagai model dasar

Berikut ini adalah contoh IAM kebijakan untuk mesin status untuk mengakses model dasar tertentu bernama `amazon.titan-text-express-v1` sebagai model dasar menggunakan tindakan CreateModelCustomizationJobAPI.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob1",
            "Action": [
                "bedrock:CreateModelCustomizationJob"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2::foundation-model/amazon.titan-text-express-
v1",
                "arn:aws:bedrock:us-east-2:123456789012:custom-model/*",
                "arn:aws:bedrock:us-east-2:123456789012:model-customization-job/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob2",
            "Action": [
```

```
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/myRole"
            ]
        }
    ]
}
```

IAMcontoh kebijakan untuk mengakses model kustom tertentu sebagai model dasar

Berikut ini adalah contoh IAM kebijakan untuk mesin status untuk mengakses model kustom tertentu sebagai model dasar menggunakan tindakan CreateModelCustomizationJobAPI.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob1",
            "Action": [
                "bedrock:CreateModelCustomizationJob"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2:123456789012:custom-model/*",
                "arn:aws:bedrock:us-east-2:123456789012:model-customization-job/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob2",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/[[roleName]]"
            ]
        }
    ]
}
```

Contoh IAM kebijakan akses penuh untuk CreateModelCustomizationJob menggunakan.sync

Berikut ini adalah contoh IAM kebijakan untuk mesin status yang menyediakan akses penuh saat Anda menggunakan tindakan CreateModelCustomizationJobAPI.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob1",
            "Action": [
                "bedrock:CreateModelCustomizationJob"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2::foundation-model/*",
                "arn:aws:bedrock:us-east-2:123456789012:custom-model/*",
                "arn:aws:bedrock:us-east-2:123456789012:model-customization-job/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob2",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/myRole"
            ]
        }
    ]
}
```

IAMcontoh kebijakan untuk mengakses model fondasi tertentu menggunakan CreateModelCustomizationJob .sync

Berikut ini adalah contoh IAM kebijakan untuk mesin status untuk mengakses model dasar tertentu bernama `amazon.titan-text-express-v1` menggunakan CreateModelCustomizationJobtindakan.sync API.

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob1",
            "Action": [
                "bedrock:CreateModelCustomizationJob"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2::foundation-model/amazon.titan-text-express-
v1",
                "arn:aws:bedrock:us-east-2:123456789012:custom-model/*",
                "arn:aws:bedrock:us-east-2:123456789012:model-customization-job/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob2",
            "Action": [
                "bedrock:GetModelCustomizationJob",
                "bedrock:StopModelCustomizationJob"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2:123456789012:model-customization-job/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob3",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/myRole"
            ]
        }
    ]
}
```

IAMcontoh kebijakan untuk mengakses model kustom CreateModelCustomizationJob menggunakan.sync

Berikut ini adalah contoh IAM kebijakan untuk mesin status untuk mengakses model kustom menggunakan CreateModelCustomizationJobtindakan.sync API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob1",
            "Action": [
                "bedrock:CreateModelCustomizationJob"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2:123456789012:custom-model/*",
                "arn:aws:bedrock:us-east-2:123456789012:model-customization-job/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob2",
            "Action": [
                "bedrock:GetModelCustomizationJob",
                "bedrock:StopModelCustomizationJob"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2:123456789012:model-customization-job/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob3",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/myRole"
            ]
        }
    ]
}
```

Contoh IAM kebijakan akses penuh untuk CreateModelCustomizationJob menggunakan.sync

Berikut ini adalah contoh IAM kebijakan untuk mesin status yang menyediakan akses penuh saat Anda menggunakan CreateModelCustomizationJobtindakan.sync API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob1",
            "Action": [
                "bedrock:CreateModelCustomizationJob"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2::foundation-model/*",
                "arn:aws:bedrock:us-east-2:123456789012:custom-model/*",
                "arn:aws:bedrock:us-east-2:123456789012:model-customization-job/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob2",
            "Action": [
                "bedrock:GetModelCustomizationJob",
                "bedrock:StopModelCustomizationJob"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2:123456789012:model-customization-job/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob3",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/myRole"
            ]
        }
    ]
}
```

Kebijakan IAM untuk AWS CodeBuild

Contoh templat berikut menunjukkan cara AWS Step Functions menghasilkan kebijakan IAM berdasarkan sumber daya dalam definisi mesin status Anda. Lihat informasi yang lebih lengkap di Kebijakan IAM untuk layanan terintegrasi dan Pola integrasi layanan.

Sumber Daya:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:sa-east-1:123456789012:StepFunctionsSample-
CodeBuildExecution1111-2222-3333-wJalrXUtnFEMI-SNSTopic-bPxRfiCYEXAMPLEKEY"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "codebuild:StartBuild",
                "codebuild:StopBuild",
                "codebuild:BatchGetBuilds",
                "codebuild:BatchGetReports"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:sa-east-1:123456789012:rule/
StepFunctionsGetEventForCodeBuildStartBuildRule"
            ],
            "Effect": "Allow"
        }
```

```
    ]
}
```

StartBuild

Sumber daya statis

Run a Job (.sync)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:StartBuild",
        "codebuild:StopBuild",
        "codebuild:BatchGetBuilds"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "events:PutTargets",
        "events:PutRule",
        "events:DescribeRule"
      ],
      "Resource": [
        "arn:aws:events:[[region]]:[[accountId]]:rule/
StepFunctionsGetEventForCodeBuildStartBuildRule"
      ]
    }
  ]
}
```

Request Response

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

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:StartBuild"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
      ]
    }
  ]
}
```

Sumber daya dinamis

Run a Job (.sync)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:StartBuild",
        "codebuild:StopBuild",
        "codebuild:BatchGetBuilds"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:*:project/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "events:PutTargets",
        "events:PutRule",
        "events:DescribeRule"
      ],
      "Resource": [
        "arn:aws:events:[[region]]:[[accountId]]:rule/
StepFunctionsGetEventForCodeBuildStartBuildRule"
```

```
      ]
    }
  ]
}
```

Request Response

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:StartBuild"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:*:project/*"
      ]
    }
  ]
}
```

`StopBuild`

Sumber daya statis

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:StopBuild"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
      ]
    }
  ]
}
```

Sumber daya dinamis

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:StopBuild"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:*:project/*"
      ]
    }
  ]
}
```

`BatchDeleteBuilds`

Sumber daya statis

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:BatchDeleteBuilds"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
      ]
    }
  ]
}
```

Sumber daya dinamis

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": [
        "codebuild:BatchDeleteBuilds"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:*:project/*"
      ]
    }
  ]
}
```

`BatchGetReports`

Sumber daya statis

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:BatchGetReports"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:[[accountId]]:report-group/[[reportName]]"
      ]
    }
  ]
}
```

Sumber daya dinamis

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:BatchGetReports"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:*:report-group/*"
      ]
    }
```

```
  ]
}
```

StartBuildBatch

Sumber daya statis

Run a Job (.sync)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:StartBuildBatch",
                "codebuild:StopBuildBatch",
                "codebuild:BatchGetBuildBatches"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:[[region]]:[[accountId]]:rule/
StepFunctionsGetEventForCodeBuildStartBuildBatchRule"
            ]
        }
    ]
}
```

Request Response

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:StartBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
            ]
        }
    ]
}
```

Sumber daya dinamis

Run a Job (.sync)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:StartBuildBatch",
                "codebuild:StopBuildBatch",
                "codebuild:BatchGetBuildBatches"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:[[region]]:[[accountId]]:rule/
StepFunctionsGetEventForCodeBuildStartBuildBatchRule"
```

```
            ]
        }
    ]
}
```

Request Response

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:StartBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/*"
            ]
        }
    ]
}
```

`StopBuildBatch`

Sumber daya statis

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:StopBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
            ]
        }
    ]
}
```

Sumber daya dinamis

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:StopBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/*"
            ]
        }
    ]
}
```

`RetryBuildBatch`

Sumber daya statis

Run a Job (.sync)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:RetryBuildBatch",
                "codebuild:StopBuildBatch",
                "codebuild:BatchGetBuildBatches"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
            ]
        }
    ]
}
```

Request Response

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:RetryBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
            ]
        }
    ]
}
```

Sumber daya dinamis

Run a Job (.sync)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:RetryBuildBatch",
                "codebuild:StopBuildBatch",
                "codebuild:BatchGetBuildBatches"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/*"
            ]
        }
    ]
}
```

Request Response

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:RetryBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/*"
            ]
        }
    ]
}
```

`DeleteBuildBatch`

Sumber daya statis

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:DeleteBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
            ]
        }
    ]
}
```

Sumber daya dinamis

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

```
            "Effect": "Allow",
            "Action": [
                "codebuild:DeleteBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/*"
            ]
        }
    ]
}
```

Kebijakan IAM untuk Amazon DynamoDB

Contoh templat berikut menunjukkan cara AWS Step Functions menghasilkan kebijakan IAM berdasarkan sumber daya dalam definisi mesin status Anda. Lihat informasi yang lebih lengkap di Kebijakan IAM untuk layanan terintegrasi dan Pola integrasi layanan.

Sumber daya statis

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:[[region]]:[[accountId]]:table/[[tableName]]"
            ]
        }
    ]
}
```

Sumber daya dinamis

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem"
            ],
            "Resource": "*"
        }
    ]
}
```

Untuk informasi selengkapnya tentang kebijakan IAM untuk semua tindakan DynamoDB API, lihat kebijakan IAM dengan DynamoDB di Panduan Pengembang Amazon DynamoDB. Selain itu, untuk informasi tentang kebijakan IAM untuk PartiQL for DynamoDB, lihat kebijakan IAM dengan PartiQL for DynamoDB di Panduan Pengembang Amazon DynamoDB.

Kebijakan IAM untuk Amazon ECS/AWS Fargate

Contoh templat berikut menunjukkan cara AWS Step Functions menghasilkan kebijakan IAM berdasarkan sumber daya dalam definisi mesin status Anda. Lihat informasi yang lebih lengkap di Kebijakan IAM untuk layanan terintegrasi dan Pola integrasi layanan.

Karena nilai untuk `TaskId` tidak diketahui sampai tugas diajukan, Step Functions membuat kebijakan `"Resource": "*"` lebih istimewa.

> **Note**
>
> Anda hanya dapat menghentikan tugas Amazon Elastic Container Service (Amazon ECS) yang dimulai oleh Step Functions, terlepas dari Kebijakan IAM `"*"`.

Run a Job (.sync)

Sumber daya statis

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

```
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RunTask"
            ],
            "Resource": [
                "arn:aws:ecs:[[region]]:
[[accountId]]:task-definition/[[taskDefinition]]"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecs:StopTask",
                "ecs:DescribeTasks"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
               "arn:aws:events:[[region]]:
[[accountId]]:rule/StepFunctionsGetEventsForECSTaskRule"
            ]
        }
    ]
}
```

Sumber daya dinamis

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "ecs:RunTask",
                "ecs:StopTask",
                "ecs:DescribeTasks"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
               "arn:aws:events:[[region]]:
[[accountId]]:rule/StepFunctionsGetEventsForECSTaskRule"
            ]
        }
    ]
}
```

Request Response and Callback (.waitForTaskToken)

Sumber daya statis

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RunTask"
            ],
            "Resource": [
                "arn:aws:ecs:[[region]]:
[[accountId]]:task-definition/[[taskDefinition]]"
            ]
        }
    ]
}
```

Sumber daya dinamis

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RunTask"
            ],
            "Resource": "*"
        }
    ]
}
```

Jika tugas Amazon ECS terjadwal Anda memerlukan penggunaan peran eksekusi tugas, peran tugas, atau penggantian peran tugas, maka Anda harus menambahkan `iam:PassRole` izin untuk setiap peran eksekusi tugas, peran tugas, atau peran tugas yang diganti ke peran IAM CloudWatch Peristiwa entitas pemanggil, yang dalam hal ini adalah Step Functions.

Kebijakan IAM untuk Amazon EKS

Contoh templat berikut menunjukkan cara AWS Step Functions menghasilkan kebijakan IAM berdasarkan sumber daya dalam definisi mesin status Anda. Lihat informasi yang lebih lengkap di Kebijakan IAM untuk layanan terintegrasi dan Pola integrasi layanan.

`CreateCluster`

Sumber Daya

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:CreateCluster"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "eks:DescribeCluster",
                "eks:DeleteCluster"
            ],
            "Resource": "arn:aws:eks:sa-east-1:444455556666:cluster/*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "arn:aws:iam::444455556666:role/StepFunctionsSample-EKSClusterManag-
EKSServiceRole-ANPAJ2UCCR6DPCEXAMPLE"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "eks.amazonaws.com"
                }
            }
        }
    ]
}
```

`CreateNodeGroup`

Sumber Daya

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeSubnets",
                "eks:CreateNodegroup"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeNodegroup",
                "eks:DeleteNodegroup"
```

```
            ],
            "Resource": "arn:aws:eks:sa-east-1:444455556666:nodegroup/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:ListAttachedRolePolicies"
            ],
            "Resource": "arn:aws:iam::444455556666:role/*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "arn:aws:iam::444455556666:role/StepFunctionsSample-EKSClusterMan-
NodeInstanceRole-ANPAJ2UCCR6DPCEXAMPLE"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "eks.amazonaws.com"
                }
            }
        }
    ]
}
```

`DeleteCluster`

Sumber Daya

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DeleteCluster",
                "eks:DescribeCluster"
            ],
            "Resource": [
                "arn:aws:eks:sa-east-1:444455556666:cluster/ExampleCluster"
            ]
        }
```

```
    ]
}
```

`DeleteNodegroup`

Sumber Daya

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DeleteNodegroup",
                "eks:DescribeNodegroup"
            ],
            "Resource": [
                "arn:aws:eks:sa-east-1:444455556666:nodegroup/ExampleCluster/
ExampleNodegroup/*"
            ]
        }
    ]
}
```

Untuk informasi selengkapnya tentang menggunakan Amazon EKS dengan Step Functions, lihat Panggil Amazon EKS dengan Step Functions.

Kebijakan IAM untuk Amazon EMR

Contoh templat berikut menunjukkan cara AWS Step Functions menghasilkan kebijakan IAM berdasarkan sumber daya dalam definisi mesin status Anda. Lihat informasi yang lebih lengkap di Kebijakan IAM untuk layanan terintegrasi dan Pola integrasi layanan.

`addStep`

Sumber daya statis

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": [
        "elasticmapreduce:AddJobFlowSteps",
        "elasticmapreduce:DescribeStep",
        "elasticmapreduce:CancelSteps"
      ],
      "Resource": [
        "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
      ]
    }
  ]
}
```

Sumber daya dinamis

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:AddJobFlowSteps",
        "elasticmapreduce:DescribeStep",
        "elasticmapreduce:CancelSteps"
      ],
      "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
    }
  ]
}
```

`cancelStep`

Sumber daya statis

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticmapreduce:CancelSteps",
            "Resource": [

 "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
            ]
```

```
        }
    ]
}
```

Sumber daya dinamis

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticmapreduce:CancelSteps",
            "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
        }
    ]
}
```

`createCluster`

Sumber daya statis

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:RunJobFlow",
        "elasticmapreduce:DescribeCluster",
        "elasticmapreduce:TerminateJobFlows"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": [
        "arn:aws:iam::{{account}}:role/[[roleName]]"
      ]
    }
  ]
}
```

`setClusterTerminationProtection`

Sumber daya statis

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticmapreduce:SetTerminationProtection",
            "Resource": [

 "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
            ]
        }
    ]
}
```

Sumber daya dinamis

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticmapreduce:SetTerminationProtection",
            "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
        }
    ]
}
```

`modifyInstanceFleetByName`

Sumber daya statis

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:ModifyInstanceFleet",
                "elasticmapreduce:ListInstanceFleets"
```

```
            ],
            "Resource": [

 "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
            ]
        }
    ]
}
```

Sumber daya dinamis

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:ModifyInstanceFleet",
                "elasticmapreduce:ListInstanceFleets"
            ],
            "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
        }
    ]
}
```

`modifyInstanceGroupByName`

Sumber daya statis

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:ModifyInstanceGroups",
                "elasticmapreduce:ListInstanceGroups"
            ],
            "Resource": [

 "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
            ]
        }
```

```
    ]
}
```

Sumber daya dinamis

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:ModifyInstanceGroups",
                "elasticmapreduce:ListInstanceGroups"
            ],
            "Resource": "*"
        }
    ]
}
```

`terminateCluster`

Sumber daya statis

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:TerminateJobFlows",
        "elasticmapreduce:DescribeCluster"
      ],
      "Resource": [
        "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
      ]
    }
  ]
}
```

Sumber daya dinamis

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

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:TerminateJobFlows",
        "elasticmapreduce:DescribeCluster"
      ],
      "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
    }
  ]
}
```

Kebijakan IAM untuk Amazon EMR di EKS

Contoh templat berikut menunjukkan cara AWS Step Functions menghasilkan kebijakan IAM berdasarkan sumber daya dalam definisi mesin status Anda. Lihat informasi yang lebih lengkap di Kebijakan IAM untuk layanan terintegrasi dan Pola integrasi layanan.

`CreateVirtualCluster`

Sumber Daya

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "emr-containers:CreateVirtualCluster"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::{{accountId}}:role/aws-service-role/emr-
containers.amazonaws.com/AnAWSServiceRoleForAmazonEMRContainers",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "emr-containers.amazonaws.com"
                }
            }
        }
```

```
    ]
}
```

`DeleteVirtualCluster`

Sumber daya statis

Run a Job (.sync)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:DeleteVirtualCluster",
        "emr-containers:DescribeVirtualCluster"
      ],
      "Resource": [
        "arn:aws:emr-containers:{{region}}:{{accountId}}:/virtualclusters/
[[virtualClusterId]]"
      ]
    }
  ]
}
```

Request Response

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:DeleteVirtualCluster"
      ],
      "Resource": [
        "arn:aws:emr-containers:{{region}}:{{accountId}}:/virtualclusters/
[[virtualClusterId]]"
      ]
    }
  ]
}
```

Sumber daya dinamis

Run a Job (.sync)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:DeleteVirtualCluster",
        "emr-containers:DescribeVirtualCluster"
      ],
      "Resource": [
        "arn:aws:emr-containers:{{region}}:{{accountId}}:/virtualclusters/*"
      ]
    }
  ]
}
```

Request Response

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:DeleteVirtualCluster"
      ],
      "Resource": [
        "arn:aws:emr-containers:{{region}}:{{accountId}}:/virtualclusters/*"
      ]
    }
  ]
}
```

`StartJobRun`

Sumber daya statis

Run a Job (.sync)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "emr-containers:StartJobRun",
      "Resource": [
        "arn:aws:emr-containers:{{region}}:{{accountId}}:/virtualclusters/
[[virtualClusterId]]"
      ],
      "Condition": {
        "StringEquals": {
          "emr-containers:ExecutionRoleArn": [
            "[[executionRoleArn]]"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:DescribeJobRun",
        "emr-containers:CancelJobRun"
      ],
      "Resource": [
        "arn:aws:emr-containers:{{region}}:{{accountId}}:/virtualclusters/
[[virtualClusterId]]/jobruns/*"
      ]
    }
  ]
}
```

Request Response

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "emr-containers:StartJobRun",
      "Resource": [
```

```
        "arn:aws:emr-containers:{{region}}:{{accountId}}:/virtualclusters/
[[virtualClusterId]]"
      ],
      "Condition": {
        "StringEquals": {
          "emr-containers:ExecutionRoleArn": [
            "[[executionRoleArn]]"
          ]
        }
      }
    }
  ]
}
```

Sumber daya dinamis

Run a Job (.sync)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "emr-containers:StartJobRun",
      "Resource": [
        "arn:aws:emr-containers:{{region}}:{{accountId}}:/virtualclusters/*"
      ],
      "Condition": {
        "StringEquals": {
          "emr-containers:ExecutionRoleArn": [
            "[[executionRoleArn]]"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:DescribeJobRun",
        "emr-containers:CancelJobRun"
      ],
      "Resource": [
```

```
        "arn:aws:emr-containers:{{region}}:{{accountId}}:/virtualclusters/*"
      ]
    }
  ]
}
```

Request Response

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "emr-containers:StartJobRun",
      "Resource": [
        "arn:aws:emr-containers:{{region}}:{{accountId}}:/virtualclusters/*"
      ],
      "Condition": {
        "StringEquals": {
          "emr-containers:ExecutionRoleArn": [
            "[[executionRoleArn]]"
          ]
        }
      }
    }
  ]
}
```

Kebijakan IAM untuk Amazon EMR Serverless

Saat Anda membuat mesin status menggunakan konsol, Step Functions secara otomatis membuat peran eksekusi untuk mesin status Anda dengan hak istimewa paling sedikit yang diperlukan. IAMPeran yang dihasilkan secara otomatis ini berlaku untuk Wilayah AWS di mana Anda membuat mesin status.

Contoh templat berikut menunjukkan cara AWS Step Functions menghasilkan kebijakan IAM berdasarkan sumber daya dalam definisi mesin status Anda. Lihat informasi yang lebih lengkap di Kebijakan IAM untuk layanan terintegrasi dan Pola integrasi layanan.

Sebaiknya saat Anda membuat IAM kebijakan, jangan sertakan wildcard dalam kebijakan. Sebagai praktik terbaik keamanan, Anda harus mencatat kebijakan Anda sebanyak mungkin. Anda harus menggunakan kebijakan dinamis hanya jika parameter input tertentu tidak diketahui selama runtime.

Selanjutnya, pengguna administrator harus berhati-hati saat memberikan peran eksekusi pengguna non-administrator untuk menjalankan mesin status. Sebaiknya sertakan kebijakan PassRole dalam peran eksekusi jika Anda membuat kebijakan sendiri. Kami juga menyarankan Anda menambahkan kunci `aws:SourceARN` dan `aws:SourceAccount` konteks dalam peran eksekusi.

Dalam topik ini:

- Contoh kebijakan IAM untuk integrasi EMR Tanpa Server dengan Step Functions

Contoh kebijakan IAM untuk integrasi EMR Tanpa Server dengan Step Functions

- Contoh kebijakan IAM untuk CreateApplication
- Contoh kebijakan IAM untuk StartApplication
- Contoh kebijakan IAM untuk StopApplication
- Contoh kebijakan IAM untuk DeleteApplication
- Contoh kebijakan IAM untuk StartJobRun
- Contoh kebijakan IAM untuk CancelJobRun

Contoh kebijakan IAM untuk CreateApplication

Berikut ini adalah contoh kebijakan IAM untuk mesin negara dengan CreateApplication Tugas status.

> **Note**
>
> Anda perlu menentukan CreateServiceLinkedRole izin dalam kebijakan IAM Anda selama pembuatan aplikasi pertama di akun Anda. Setelah itu, Anda tidak perlu menambahkan izin ini. Untuk informasi tentang CreateServiceLinkedRole, lihat CreateServiceLinkedRoledi https://docs.aws.amazon.com/IAM/latest/APIReference/.

Sumber daya statis dan dinamis untuk kebijakan berikut adalah sama.

Run a Job (.sync)

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "emr-serverless:CreateApplication"
            ],
            "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "emr-serverless:GetApplication",
                "emr-serverless:DeleteApplication"
            ],
            "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:{{region}}:{{accountId}}:rule/
StepFunctionsGetEventsForEMRServerlessApplicationRule"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::{{accountId}}:role/aws-service-role/ops.emr-
serverless.amazonaws.com/AWS ServiceRoleForAmazonEMRServerless*",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "ops.emr-serverless.amazonaws.com"
```

```
                }
            }
        }
    ]
}
```

Request Response

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

   "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "emr-serverless:CreateApplication"
            ],
            "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::{{accountId}}:role/aws-service-role/ops.emr-
serverless.amazonaws.com/AWS ServiceRoleForAmazonEMRServerless*",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "ops.emr-serverless.amazonaws.com"
                }
            }
        }
    ]
}
```

Contoh kebijakan IAM untuk StartApplication

Sumber daya statis

Berikut ini adalah contoh kebijakan IAM untuk sumber daya statis saat Anda menggunakan mesin status dengan StartApplication Tugas status.

Run a Job (.sync)

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:StartApplication",
               "emr-serverless:GetApplication",
               "emr-serverless:StopApplication"
            ],
           "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/
[[applicationId]]"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:{{region}}:
{{accountId}}:rule/StepFunctionsGetEventsForEMRServerlessApplicationRule"
            ]
        }
    ]
}
```

Request Response

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:StartApplication"
            ],
           "Resource": [
```

```
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/
[[applicationId]]"
            ]
        }
    ]
}
```

Sumber daya dinamis

Berikut ini adalah contoh kebijakan IAM untuk sumber daya dinamis saat Anda menggunakan mesin status dengan StartApplication Tugas status.

Run a Job (.sync)

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:StartApplication",
               "emr-serverless:GetApplication",
               "emr-serverless:StopApplication"
            ],
           "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:{{region}}:
{{accountId}}:rule/StepFunctionsGetEventsForEMRServerlessApplicationRule"
            ]
        }
    ]
}
```

Request Response

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:StartApplication"
            ],
           "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/*"
            ]
        }
    ]
}
```

Contoh kebijakan IAM untuk StopApplication

Sumber daya statis

Berikut ini adalah contoh kebijakan IAM untuk sumber daya statis saat Anda menggunakan mesin status dengan StopApplication Tugas status.

Run a Job (.sync)

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:StopApplication",
               "emr-serverless:GetApplication"
            ],
           "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/
[[applicationId]]"
            ]
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:{{region}}:
{{accountId}}:rule/StepFunctionsGetEventsForEMRServerlessApplicationRule"
            ]
        }
    ]
}
```

Request Response

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:StopApplication"
            ],
           "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/
[[applicationId]]"
            ]
        }
    ]
}
```

Sumber daya dinamis

Berikut ini adalah contoh kebijakan IAM untuk sumber daya dinamis saat Anda menggunakan mesin status dengan StopApplication Tugas status.

Run a Job (.sync)

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

```
            "Effect": "Allow",
            "Action": [
                "emr-serverless:StopApplication",
                "emr-serverless:GetApplication"
             ],
            "Resource": [
                 "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/*"
             ]
        },
        {
             "Effect": "Allow",
             "Action": [
                 "events:PutTargets",
                 "events:PutRule",
                 "events:DescribeRule"
             ],
             "Resource": [
                 "arn:aws:events:{{region}}:
{{accountId}}:rule/StepFunctionsGetEventsForEMRServerlessApplicationRule"
             ]
        }
    ]
}
```

Request Response

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "emr-serverless:StopApplication"
             ],
            "Resource": [
                 "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/*"
             ]
        }
    ]
}
```

Contoh kebijakan IAM untuk DeleteApplication

Sumber daya statis

Berikut ini adalah contoh kebijakan IAM untuk sumber daya statis saat Anda menggunakan mesin status dengan DeleteApplication Tugas status.

Run a Job (.sync)

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:DeleteApplication",
               "emr-serverless:GetApplication"
            ],
           "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/
[[applicationId]]"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:{{region}}:
{{accountId}}:rule/StepFunctionsGetEventsForEMRServerlessApplicationRule"
            ]
        }
    ]
}
```

Request Response

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

```
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:DeleteApplication"
            ],
           "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/
[[applicationId]]"
            ]
        }
    ]
}
```

Sumber daya dinamis

Berikut ini adalah contoh kebijakan IAM untuk sumber daya dinamis saat Anda menggunakan mesin status dengan DeleteApplication Tugas status.

Run a Job (.sync)

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:DeleteApplication",
               "emr-serverless:GetApplication"
            ],
           "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
```

```
                "arn:aws:events:{{region}}:
{{accountId}}:rule/StepFunctionsGetEventsForEMRServerlessApplicationRule"
            ]
        }
    ]
}
```

Request Response

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:DeleteApplication"
            ],
           "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/*"
            ]
        }
    ]
}
```

Contoh kebijakan IAM untuk StartJobRun

Sumber daya statis

Berikut ini adalah contoh kebijakan IAM untuk sumber daya statis saat Anda menggunakan mesin status dengan StartJobRun Tugas status.

Run a Job (.sync)

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:StartJobRun"
            ],
           "Resource": [
```

```
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/
[[applicationId]]"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "[[jobExecutionRoleArn]]"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "emr-serverless.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "emr-serverless:GetJobRun",
                "emr-serverless:CancelJobRun"
            ],
            "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/
[[applicationId]]/jobruns/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:{{region}}:
{{accountId}}:rule/StepFunctionsGetEventsForEMRServerlessJobRule"
            ]
        }
    ]
}
```

Request Response

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:StartJobRun"
            ],
           "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/
[[applicationId]]"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "[[jobExecutionRoleArn]]"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "emr-serverless.amazonaws.com"
                }
            }
        }
    ]
}
```

Sumber daya dinamis

Berikut ini adalah contoh kebijakan IAM untuk sumber daya dinamis saat Anda menggunakan mesin status dengan StartJobRun Tugas status.

Run a Job (.sync)

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
```

```
            "Action": [
                "emr-serverless:StartJobRun",
                "emr-serverless:GetJobRun",
                "emr-serverless:CancelJobRun"
            ],
            "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "[[jobExecutionRoleArn]]"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "emr-serverless.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:{{region}}:
{{accountId}}:rule/StepFunctionsGetEventsForEMRServerlessJobRule"
            ]
        }
    ]
}
```

Request Response

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
```

```
            "Action": [
                "emr-serverless:StartJobRun"
             ],
            "Resource": [
                 "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/*"
             ]
        },
        {
             "Effect": "Allow",
             "Action": "iam:PassRole",
             "Resource": [
                 "[[jobExecutionRoleArn]]"
             ],
             "Condition": {
                 "StringEquals": {
                     "iam:PassedToService": "emr-serverless.amazonaws.com"
                 }
             }
        }
    ]
}
```

Contoh kebijakan IAM untuk CancelJobRun

Sumber daya statis

Berikut ini adalah contoh kebijakan IAM untuk sumber daya statis saat Anda menggunakan mesin status dengan CancelJobRun Tugas status.

Run a Job (.sync)

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "emr-serverless:CancelJobRun",
                "emr-serverless:GetJobRun"
             ],
            "Resource": [
                 "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/
[[applicationId]]/jobruns/[[jobRunId]]"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:{{region}}:
{{accountId}}:rule/StepFunctionsGetEventsForEMRServerlessJobRule"
            ]
        }
    ]
}
```

Request Response

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:CancelJobRun"
            ],
           "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/
[[applicationId]]/jobruns/[[jobRunId]]"
            ]
        }
    ]
}
```

Sumber daya dinamis

Berikut ini adalah contoh kebijakan IAM untuk sumber daya dinamis saat Anda menggunakan mesin status dengan CancelJobRun Tugas status.

Run a Job (.sync)

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:CancelJobRun",
               "emr-serverless:GetJobRun"
            ],
           "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:{{region}}:
{{accountId}}:rule/StepFunctionsGetEventsForEMRServerlessJobRule"
            ]
        }
    ]
}
```

Request Response

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:CancelJobRun"
            ],
           "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/*"
            ]
```

```
        }
    ]
}
```

Kebijakan IAM untuk Amazon EventBridge

Contoh templat berikut menunjukkan cara AWS Step Functions menghasilkan kebijakan IAM berdasarkan sumber daya dalam definisi mesin status Anda. Lihat informasi yang lebih lengkap di Kebijakan IAM untuk layanan terintegrasi dan Pola integrasi layanan.

`PutEvents`

Sumber daya statis

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "events:PutEvents"
            ],
            "Resource": [
                "arn:aws:events:us-east-1:123456789012:event-bus/stepfunctions-
sampleproject-eventbus"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Sumber daya dinamis

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "events:PutEvents"
            ],
            "Resource": "arn:aws:events:*:*:event-bus/*"
        }
```

```
    ]
}
```

Untuk informasi selengkapnya tentang penggunaan EventBridge dengan Step Functions, lihatPanggilan EventBridge dengan Step Functions.

Kebijakan IAM untuk AWS Lambda

Contoh templat berikut menunjukkan cara AWS Step Functions menghasilkan kebijakan IAM berdasarkan sumber daya dalam definisi mesin status Anda. Lihat informasi yang lebih lengkap di Kebijakan IAM untuk layanan terintegrasi dan Pola integrasi layanan.

AWS Step Functions menghasilkan kebijakan IAM berdasarkan definisi mesin negara Anda. Untuk mesin status dengan dua status AWS Lambda tugas yang memanggil `function1` dan`function2`, kebijakan dengan `lambda:Invoke` izin untuk dua fungsi harus digunakan.

Seperti yang ditunjukkan dalam contoh berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:[[region]]:[[accountId]]:function:[[function1]]",
                "arn:aws:lambda:[[region]]:[[accountId]]:function:[[function2]]"
            ]
        }
    ]
}
```

Kebijakan IAM untuk AWS Elemental MediaConvert

Contoh templat berikut menunjukkan bagaimana AWS Step Functions Anda harus menyiapkan kebijakan IAM berdasarkan sumber daya dalam definisi mesin status Anda. Anda dapat menggunakan konsol IAM untuk menambahkan kebijakan peran yang hilang. Lihat informasi yang lebih lengkap di Kebijakan IAM untuk layanan terintegrasi dan Pola integrasi layanan.

Karena MediaConvert memberikan dukungan sebagian untuk kontrol akses tingkat sumber daya, Anda harus menggunakannya. `"Resource": "*"`

Run a Job (.sync)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "mediaconvert:CreateJob",
                "mediaconvert:GetJob",
                "mediaconvert:CancelJob"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
               "arn:aws:events:[[region]]:[[accountId]]:rule/
StepFunctionsGetEventsForMediaConvertJobRule"
            ]
        }
    ]
}
```

Request Response

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

```
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "mediaconvert:CreateJob"
            ],
            "Resource": "*"
        }
    ]
}
```

Kebijakan IAM untuk AWS Glue

Contoh templat berikut menunjukkan cara AWS Step Functions menghasilkan kebijakan IAM berdasarkan sumber daya dalam definisi mesin status Anda. Lihat informasi yang lebih lengkap di Kebijakan IAM untuk layanan terintegrasi dan Pola integrasi layanan.

AWS Glue tidak memiliki kontrol berbasis sumber daya.

Run a Job (.sync)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:StartJobRun",
                "glue:GetJobRun",
                "glue:GetJobRuns",
                "glue:BatchStopJobRun"
            ],
            "Resource": "*"
        }
    ]
}
```

Request Response and Callback (.waitForTaskToken)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:StartJobRun"
            ],
            "Resource": "*"
        }
    ]
}
```

Kebijakan IAM untuk AWS Glue DataBrew

Contoh templat berikut menunjukkan cara AWS Step Functions menghasilkan kebijakan IAM berdasarkan sumber daya dalam definisi mesin status Anda. Lihat informasi yang lebih lengkap di Kebijakan IAM untuk layanan terintegrasi dan Pola integrasi layanan.

Run a Job (.sync)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "databrew:startJobRun",
                "databrew:listJobRuns",
                "databrew:stopJobRun"
            ],
            "Resource": [
                "arn:aws:databrew:{{region}}:{{accountId}}:job/*"
            ]
        }
    ]
}
```

Request Response

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "databrew:startJobRun"
            ],
            "Resource": [
                "arn:aws:databrew:{{region}}:{{accountId}}:job/*"
            ]
        }
    ]
}
```

Kebijakan IAM untuk Amazon SageMaker

Contoh templat berikut menunjukkan cara AWS Step Functions menghasilkan kebijakan IAM berdasarkan sumber daya dalam definisi mesin status Anda. Lihat informasi yang lebih lengkap di Kebijakan IAM untuk layanan terintegrasi dan Pola integrasi layanan.

> **Note**
>
> Untuk contoh ini, *[[roleArn]]* mengacu pada Amazon Resource Name (ARN) dari peran IAM yang SageMaker digunakan untuk mengakses artefak model dan gambar docker untuk penerapan pada instance komputasi HTML, atau untuk pekerjaan transformasi batch. Untuk informasi selengkapnya, lihat SageMaker Peran Amazon.

`CreateTrainingJob`

Sumber daya statis

Run a Job (.sync)

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

```
      "Effect": "Allow",
      "Action": [
        "sagemaker:CreateTrainingJob",
        "sagemaker:DescribeTrainingJob",
        "sagemaker:StopTrainingJob"
      ],
      "Resource": [
        "arn:aws:sagemaker:[[region]]:[[accountId]]:training-
job/[[trainingJobName]]*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:ListTags"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "[[roleArn]]"
      ],
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "sagemaker.amazonaws.com"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "events:PutTargets",
        "events:PutRule",
        "events:DescribeRule"
      ],
      "Resource": [
        "arn:aws:events:[[region]]:[[accountId]]:rule/
StepFunctionsGetEventsForSageMakerTrainingJobsRule"
```

```
      ]
    }
  ]
}
```

Request Response and Callback (.waitForTaskToken)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:CreateTrainingJob"
      ],
      "Resource": [
        "arn:aws:sagemaker:[[region]]:[[accountId]]:training-
job/[[trainingJobName]]*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:ListTags"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "[[roleArn]]"
      ],
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "sagemaker.amazonaws.com"
        }
      }
```

```
    }
  ]
}
```

Sumber daya dinamis

.sync or .waitForTaskToken

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:CreateTrainingJob",
        "sagemaker:DescribeTrainingJob",
        "sagemaker:StopTrainingJob"
      ],
      "Resource": [
        "arn:aws:sagemaker:[[region]]:[[accountId]]:training-job/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:ListTags"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "[[roleArn]]"
      ],
      "Condition": {
        "StringEquals": {
```

```
          "iam:PassedToService": "sagemaker.amazonaws.com"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "events:PutTargets",
        "events:PutRule",
        "events:DescribeRule"
      ],
      "Resource": [
        "arn:aws:events:[[region]]:[[accountId]]:rule/
StepFunctionsGetEventsForSageMakerTrainingJobsRule"
      ]
    }
  ]
}
```

Request Response and Callback (.waitForTaskToken)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:CreateTrainingJob"
      ],
      "Resource": [
        "arn:aws:sagemaker:[[region]]:[[accountId]]:training-job/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:ListTags"
      ],
      "Resource": [
        "*"
      ]
    },
    {
```

```
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "[[roleArn]]"
      ],
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "sagemaker.amazonaws.com"
        }
      }
    }
  ]
}
```

`CreateTransformJob`

> **Note**
>
> AWS Step Functions tidak akan secara otomatis membuat kebijakan `CreateTransformJob` saat Anda membuat mesin status yang terintegrasi dengannya SageMaker. Anda harus melampirkan kebijakan sebaris untuk peran yang dibuat berdasarkan salah satu contoh IAM berikut.

Sumber daya statis

Run a Job (.sync)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:CreateTransformJob",
        "sagemaker:DescribeTransformJob",
        "sagemaker:StopTransformJob"
      ],
      "Resource": [
```

```
        "arn:aws:sagemaker:[[region]]:[[accountId]]:transform-
job/[[transformJobName]]*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:ListTags"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "[[roleArn]]"
      ],
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "sagemaker.amazonaws.com"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "events:PutTargets",
        "events:PutRule",
        "events:DescribeRule"
      ],
      "Resource": [
        "arn:aws:events:[[region]]:[[accountId]]:rule/
StepFunctionsGetEventsForSageMakerTransformJobsRule"
      ]
    }
  ]
}
```

Request Response and Callback (.waitForTaskToken)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:CreateTransformJob"
      ],
      "Resource": [
        "arn:aws:sagemaker:[[region]]:[[accountId]]:transform-
job/[[transformJobName]]*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:ListTags"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "[[roleArn]]"
      ],
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "sagemaker.amazonaws.com"
        }
      }
    }
  ]
}
```

Sumber daya dinamis

Run a Job (.sync)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:CreateTransformJob",
        "sagemaker:DescribeTransformJob",
        "sagemaker:StopTransformJob"
      ],
      "Resource": [
        "arn:aws:sagemaker:[[region]]:[[accountId]]:transform-job/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:ListTags"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "[[roleArn]]"
      ],
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "sagemaker.amazonaws.com"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
        "events:PutTargets",
        "events:PutRule",
        "events:DescribeRule"
      ],
      "Resource": [
        "arn:aws:events:[[region]]:[[accountId]]:rule/
StepFunctionsGetEventsForSageMakerTransformJobsRule"
      ]
    }
  ]
}
```

Request Response and Callback (.waitForTaskToken)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:CreateTransformJob"
      ],
      "Resource": [
        "arn:aws:sagemaker:[[region]]:[[accountId]]:transform-job/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:ListTags"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "[[roleArn]]"
```

```
      ],
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "sagemaker.amazonaws.com"
        }
      }
    }
  ]
}
```

Kebijakan IAM untuk Amazon SNS

Contoh templat berikut menunjukkan cara AWS Step Functions menghasilkan kebijakan IAM berdasarkan sumber daya dalam definisi mesin status Anda. Lihat informasi yang lebih lengkap di Kebijakan IAM untuk layanan terintegrasi dan Pola integrasi layanan.

Sumber daya statis

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:[[region]]:[[accountId]]:[[topicName]]"
            ]
        }
    ]
}
```

Sumber daya berdasarkan Path, atau penerbitan ke *TargetArn* atau *PhoneNumber*

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

```
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": "*"
        }
    ]
}
```

Kebijakan IAM untuk Amazon SQS

Contoh templat berikut menunjukkan cara AWS Step Functions menghasilkan kebijakan IAM berdasarkan sumber daya dalam definisi mesin status Anda. Lihat informasi yang lebih lengkap di Kebijakan IAM untuk layanan terintegrasi dan Pola integrasi layanan.

Sumber daya statis

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sqs:SendMessage"
            ],
            "Resource": [
                "arn:aws:sqs:[[region]]:[[accountId]]:[[queueName]]"
            ]
        }
    ]
}
```

Sumber daya dinamis

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sqs:SendMessage"
```

```
            ],
            "Resource": "*"
        }
    ]
}
```

Kebijakan IAM untuk AWS Step Functions

Untuk mesin status yang memanggil `StartExecution` untuk eksekusi alur kerja bersarang tunggal, gunakan kebijakan IAM yang membatasi izin ke mesin status tersebut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "states:StartExecution"
            ],
            "Resource": [

 "arn:aws:states:[[region]]:[[accountId]]:stateMachine:[[stateMachineName]]"
            ]
        }
    ]
}
```

Untuk informasi selengkapnya, lihat hal berikut:

- Menggunakan AWS Step Functions dengan layanan lain
- Meneruskan parameter ke API layanan
- Kelola AWS Step Functions Eksekusi sebagai Layanan Terpadu

Synchronous

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

```
            "Effect": "Allow",
            "Action": [
                "states:StartExecution"
            ],
            "Resource": [
                "arn:aws:states:[[region]]:[[accountId]]:stateMachine:
[[stateMachineName]]"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "states:DescribeExecution",
                "states:StopExecution"
            ],
            "Resource": [

 "arn:aws:states:[[region]]:[[accountId]]:execution:[[stateMachineName]]:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
               "arn:aws:events:[[region]]:[[accountId]]:rule/
StepFunctionsGetEventsForStepFunctionsExecutionRule"
            ]
        }
    ]
}
```

Asynchronous

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "states:StartExecution"
            ],
            "Resource": [

 "arn:aws:states:[[region]]:[[accountId]]:stateMachine:[[stateMachineName]]"
            ]
        }
    ]
}
```

Untuk informasi selengkapnya tentang eksekusi alur kerja bersarang, lihat Mulai Eksekusi Alur Kerja dari Status Tugas..

Kebijakan IAM untuk AWS X-Ray

Contoh templat berikut menunjukkan cara AWS Step Functions menghasilkan kebijakan IAM berdasarkan sumber daya dalam definisi mesin status Anda. Lihat informasi yang lebih lengkap di Kebijakan IAM untuk layanan terintegrasi dan Pola integrasi layanan.

Untuk mengaktifkan pelacakan X-Ray, Anda memerlukan kebijakan IAM dengan izin yang sesuai untuk mengizinkan pelacakan. Jika mesin status Anda menggunakan layanan terintegrasi lainnya, Anda mungkin memerlukan kebijakan IAM tambahan. Lihat kebijakan IAM untuk integrasi layanan khusus.

Ketika Anda membuat mesin status dengan pelacakan X-Ray diaktifkan, kebijakan IAM secara otomatis dibuat.

> **Note**
>
> Jika Anda mengaktifkan pelacakan X-Ray untuk mesin status yang ada, Anda harus memastikan bahwa Anda menambahkan kebijakan dengan izin yang memadai untuk mengaktifkan jejak X-Ray.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "xray:PutTraceSegments",
                "xray:PutTelemetryRecords",
                "xray:GetSamplingRules",
                "xray:GetSamplingTargets"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Untuk informasi selengkapnya tentang menggunakan X-Ray dengan Step Functions, lihat AWS X-Ray dan Step Functions.

Aktivitas atau Tidak Ada Tugas

Untuk mesin status yang hanya memiliki `Activity` tugas, atau tidak ada tugas sama sekali, gunakan kebijakan IAM yang menolak akses ke semua tindakan dan sumber daya.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "*",
            "Resource": "*"
        }
    ]
}
```

Untuk informasi selengkapnya tentang penggunaan tugas `Activity` , lihat Aktivitas.

Kebijakan IAM untuk menggunakan status Peta Terdistribusi

Saat Anda membuat alur kerja dengan konsol Step Functions, Step Functions dapat secara otomatis menghasilkan kebijakan IAM berdasarkan sumber daya dalam definisi alur kerja Anda. Kebijakan ini mencakup hak istimewa paling sedikit yang diperlukan untuk memungkinkan peran mesin status menjalankan tindakan `StartExecution` API untuk status Peta Terdistribusi. Kebijakan ini juga mencakup hak istimewa paling sedikit Step Functions yang diperlukan untuk mengakses AWS

sumber daya, seperti bucket dan objek Amazon S3 serta fungsi Lambda. Kami sangat menyarankan agar Anda hanya menyertakan izin yang diperlukan dalam kebijakan IAM Anda. Misalnya, jika alur kerja Anda menyertakan `Map` status dalam mode Terdistribusi, cakupkan kebijakan Anda ke bucket Amazon S3 tertentu dan folder yang berisi kumpulan data Anda.

> **Important**
>
> Jika Anda menentukan bucket dan objek Amazon S3, atau awalan, dengan jalur referensi ke pasangan nilai kunci yang ada di input status Peta Terdistribusi, pastikan Anda memperbarui kebijakan IAM untuk alur kerja Anda. Cakupan kebijakan hingga ke bucket dan nama objek yang diselesaikan jalur saat runtime.

Dalam topik ini:

- Contoh kebijakan IAM untuk menjalankan status Peta Terdistribusi
- Contoh kebijakan IAM untuk Peta redriving Terdistribusi
- Contoh kebijakan IAM untuk membaca data dari kumpulan data Amazon S3
- Contoh kebijakan IAM untuk menulis data ke bucket Amazon S3

Contoh kebijakan IAM untuk menjalankan status Peta Terdistribusi

Bila Anda menyertakan status Peta Terdistribusi dalam alur kerja Anda, Step Functions memerlukan izin yang sesuai untuk memungkinkan peran mesin status menjalankan tindakan `StartExecution` API untuk status Peta Terdistribusi.

Contoh kebijakan IAM berikut memberikan hak istimewa paling sedikit yang diperlukan untuk peran mesin status Anda untuk menjalankan status Peta Terdistribusi.

> **Note**
>
> Pastikan Anda mengganti *`stateMachineName`* dengan nama mesin status tempat Anda menggunakan status Peta Terdistribusi. Misalnya, `arn:aws:states:`*`us-east-2`*`:`*`123456789012`*`:stateMachine:`*`mystateMachine`*.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution"
      ],
      "Resource": [
        "arn:aws:states:region:accountID:stateMachine:stateMachineName"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeExecution",
        "states:StopExecution"
      ],
      "Resource": "arn:aws:states:region:accountID:execution:stateMachineName:*"
    }
  ]
}
```

Contoh kebijakan IAM untuk Peta redriving Terdistribusi

Anda dapat memulai ulang eksekusi alur kerja anak yang gagal di Map Run oleh alur kerja induk redrivingAnda. Alur kerja redriven induk redrives semua status gagal, termasuk Peta Terdistribusi. Pastikan peran eksekusi Anda memiliki hak istimewa paling sedikit yang diperlukan untuk memungkinkannya menjalankan tindakan `RedriveExecution` API pada alur kerja induk.

Contoh kebijakan IAM berikut memberikan hak istimewa paling sedikit yang diperlukan untuk peran mesin status Anda untuk redriving status Peta Terdistribusi.

> **Note**
>
> Pastikan Anda mengganti *`stateMachineName`* dengan nama mesin status tempat Anda menggunakan status Peta Terdistribusi. Misalnya, `arn:aws:states:`*`us-east-2`*`:`*`123456789012`*`:stateMachine:`*`mystateMachine`*.

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

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:RedriveExecution"
      ],
      "Resource": "arn:aws:states:us-
east-2:123456789012:execution:myStateMachine/myMapRunLabel:*"
    }
  ]
}
```

Contoh kebijakan IAM untuk membaca data dari kumpulan data Amazon S3

Contoh kebijakan IAM berikut memberikan hak istimewa paling sedikit yang diperlukan untuk mengakses kumpulan data Amazon S3 Anda menggunakan ListObjects tindakan V2 dan API. GetObject

Example Kebijakan IAM untuk objek Amazon S3 sebagai kumpulan data

Contoh berikut menunjukkan kebijakan IAM yang memberikan hak istimewa paling sedikit untuk mengakses objek yang terorganisir *processImages* dalam bucket Amazon S3 bernama. *myBucket*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket"
            ],
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
                        "processImages"
                    ]
                }
            }
        }
    ]
```

```
}
```

Example Kebijakan IAM untuk file CSV sebagai kumpulan data

Contoh berikut menunjukkan kebijakan IAM yang memberikan hak istimewa paling sedikit untuk mengakses file CSV bernama. *ratings.csv*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket/csvDataset/ratings.csv"
            ]
        }
    ]
}
```

Example Kebijakan IAM untuk inventaris Amazon S3 sebagai kumpulan data

Contoh berikut menunjukkan kebijakan IAM yang memberikan hak istimewa paling sedikit untuk mengakses laporan inventaris Amazon S3.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::destination-prefix/source-bucket/config-ID/YYYY-MM-DDTHH-
MMZ/manifest.json",
                "arn:aws:s3:::destination-prefix/source-bucket/config-ID/data/*"
            ]
        }
    ]
```

```
}
```

Contoh kebijakan IAM untuk menulis data ke bucket Amazon S3

Contoh kebijakan IAM berikut memberikan hak istimewa paling sedikit yang diperlukan untuk menulis hasil eksekusi alur kerja turunan Anda ke folder bernama *CSVjobs di bucket* Amazon S3 menggunakan tindakan API. PutObject

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
            "Resource": [
                "arn:aws:s3:::resultBucket/csvJobs/*"
            ]
        }
    ]
}
```

Izin IAM untuk bucket AWS KMS key Amazon S3 terenkripsi

Status Peta Terdistribusi menggunakan unggahan multibagian untuk menulis hasil eksekusi alur kerja turunan ke bucket Amazon S3. Jika bucket dienkripsi menggunakan AWS Key Management Service (AWS KMS) kunci, Anda juga harus menyertakan izin dalam IAM kebijakan untuk melakukan `kms:Decryptkms:Encrypt`, dan `kms:GenerateDataKey` tindakan pada kunci tersebut. Izin ini diperlukan karena Amazon S3 harus mendekripsi dan membaca data dari bagian file terenkripsi sebelum menyelesaikan unggahan multibagian.

Contoh kebijakan IAM berikut memberikan izin kek`ms:Decrypt`,`kms:Encrypt`, dan `kms:GenerateDataKey` tindakan pada kunci yang digunakan untuk mengenkripsi bucket Amazon S3 Anda.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:Decrypt",
      "kms:Encrypt",
      "kms:GenerateDataKey"
    ],
    "Resource": [
      "arn:aws:kms:us-east-1:123456789012:key/111aa2bb-333c-4d44-5555-a111bb2c33dd"
    ]
  }
}
```

Untuk informasi lebih lanjut, lihat Mengunggah sebuah file besar ke Amazon S3 dengan enkripsi menggunakan AWS KMS key dalam AWS Pusat Pengetahuan.

Jika pengguna atau peran IAM Anda Akun AWS sama denganKMS key, maka Anda harus memiliki izin ini pada kebijakan utama. Jika pengguna atau peran IAM Anda milik akun yang berbeda dari akunKMS key, maka Anda harus memiliki izin pada kebijakan utama dan pengguna atau peran IAM Anda.

Kebijakan berbasis tanda

Step Functions mendukung kebijakan berbasis tanda. Misalnya, Anda dapat membatasi akses ke sumber daya Step Functions yang menyertakan tanda dengan kunci `environment` dan nilai `production`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "states:TagResource",
                "states:UntagResource",
                "states:DeleteActivity",
                "states:DeleteStateMachine",
                "states:StopExecution"
            ],
            "Resource": "*",
            "Condition": {
```

```
                "StringEquals": {"aws:ResourceTag/environment": "production"}
            }
        }
    ]
}
```

Kebijakan ini akan `Deny` kemampuan untuk menghapus mesin status atau aktivitas, menghentikan eksekusi, dan menambah atau menghapus tanda baru untuk semua sumber daya yang telah ditandai sebagai `environment/production`.

Untuk otorisasi berbasis tag, sumber daya eksekusi mesin status seperti yang ditunjukkan pada contoh berikut mewarisi tag yang terkait dengan mesin status.

```
arn:<partition>:states:<Region>:<account-id>:execution:<StateMachineName>:<ExecutionId>
```

Saat Anda memanggil DescribeExecutionatau API lain di mana Anda menentukan ARN sumber daya eksekusi, Step Functions menggunakan tag yang terkait dengan mesin status untuk menerima atau menolak permintaan saat melakukan otorisasi berbasis tag. Ini membantu Anda mengizinkan atau menolak akses ke eksekusi mesin status di tingkat mesin negara.

Untuk informasi selengkapnya tentang penandaan, lihat berikut ini:

- Penandaan di Step Functions
- Mengontrol Akses Menggunakan Tag IAM

Memecahkan masalah AWS Step Functions identitas dan akses

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan Step Functions dan IAM.

Topik

- Saya tidak berwenang untuk melakukan tindakan di Step Functions
- Saya tidak berwenang untuk melakukan iam: PassRole
- Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses sumber daya Step Functions saya

Saya tidak berwenang untuk melakukan tindakan di Step Functions

Jika Anda menerima pesan kesalahan bahwa Anda tidak memiliki otorisasi untuk melakukan tindakan, kebijakan Anda harus diperbarui agar Anda dapat melakukan tindakan tersebut.

Contoh kesalahan berikut terjadi ketika pengguna `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya *`my-example-widget`* fiktif, tetapi tidak memiliki izin `states:`*`GetWidget`* fiktif.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 states:GetWidget on resource: my-example-widget
```

Dalam hal ini, kebijakan Mateo harus diperbarui untuk memungkinkannya mengakses *`my-example-widget`* sumber daya menggunakan `states:`*`GetWidget`* tindakan.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan yang tidak diizinkan untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran ke Step Functions.

Beberapa Layanan AWS memungkinkan Anda untuk meneruskan peran yang ada ke layanan tersebut 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 untuk melakukan tindakan di Step Functions. Namun, tindakan tersebut memerlukan layanan untuk mendapatkan izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut pada layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dalam kasus ini, kebijakan Mary harus diperbarui agar dia mendapatkan izin untuk melakukan tindakan `iam:PassRole` tersebut.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses sumber daya Step Functions saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau pengguna 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 (ACL), Anda dapat menggunakan kebijakan tersebut untuk memberi pengguna akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa hal berikut:

- Untuk mengetahui apakah Step Functions mendukung fitur-fitur ini, lihatBagaimana AWS Step Functions 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 (gabungan identitas) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara penggunaan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Perbedaan antara peran IAM dan kebijakan berbasis sumber daya di Panduan Pengguna IAM.

Pencatatan dan Pemantauan

Untuk informasi tentang pencatatan dan pemantauan AWS Step Functions, lihat Pencatatan dan pemantauan bagian.

Validasi Kepatuhan untuk AWS Step Functions

Auditor pihak ketiga menilai keamanan dan kepatuhan AWS Step Functions sebagai bagian dari beberapa program AWS kepatuhan. Program ini mencakup SOC, PCI, FedRAMP, HIPAA, dan lainnya.

Untuk daftar AWS layanan dalam lingkup program kepatuhan tertentu, lihat AWS Layanan dalam Lingkup oleh AWS Layanan Program Kepatuhan . 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 Step Functions ditentukan oleh sensitivitas pada data Anda, tujuan kepatuhan perusahaan Anda, serta hukum dan peraturan yang berlaku. AWS menyediakan sumber daya berikut untuk membantu kepatuhan:

- Panduan Memulai Cepat Keamanan dan Kepatuhan Panduan Keamanan dan Kepatuhan — Panduan penerapan ini membahas pertimbangan arsitektur dan memberikan langkah-langkah untuk menerapkan lingkungan dasar yang berfokus pada keamanan dan kepatuhan. AWS
- Arsitektur untuk Keamanan dan Kepatuhan HIPAA di Amazon Web Services — Whitepaper ini menjelaskan bagaimana perusahaan dapat menggunakan AWS untuk membuat aplikasi yang sesuai dengan HIPAA.
- AWS Sumber Daya AWS — Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- Mengevaluasi Sumber Daya dengan Aturan dalam Panduan AWS Config Pengembang — AWS Config Layanan menilai seberapa baik konfigurasi sumber daya Anda mematuhi praktik internal, pedoman industri, dan peraturan.
- AWS Security Hub— AWS Layanan ini memberikan pandangan komprehensif tentang keadaan keamanan Anda di dalamnya AWS yang membantu Anda memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik.

Ketahanan di AWS Step Functions

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 Availability Zone, Anda dapat mendesain dan mengoperasikan aplikasi dan basis data yang secara otomatis mengalami kegagalan 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.

Selain infrastruktur AWS global, Step Functions menawarkan beberapa fitur untuk membantu mendukung ketahanan data dan kebutuhan pencadangan Anda.

Keamanan Infrastruktur di AWS Step Functions

Sebagai layanan terkelola, AWS Step Functions 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 Step Functions 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.

Selain itu, permintaan harus ditandatangani dengan menggunakan ID kunci akses dan kunci akses rahasia yang terkait dengan pengguna utama IAM. Atau Anda dapat menggunakan AWS Security Token Service (AWS STS) untuk menghasilkan kredensial keamanan sementara untuk menandatangani permintaan.

Anda dapat memanggil operasi AWS API dari lokasi jaringan mana pun, tetapi Step Functions tidak mendukung kebijakan akses berbasis sumber daya, yang dapat mencakup pembatasan berdasarkan alamat IP sumber. Anda juga dapat menggunakan kebijakan Step Functions untuk mengontrol akses dari titik akhir Amazon Virtual Private Cloud (Amazon VPC) tertentu atau VPC tertentu. Secara efektif, ini mengisolasi akses jaringan ke Step Functions sumber daya tertentu hanya dari VPC tertentu dalam AWS jaringan.

Analisis Konfigurasi dan Kerentanan di AWS Step Functions

Konfigurasi dan kontrol TI adalah tanggung jawab bersama antara AWS dan Anda, pelanggan kami. Untuk informasi selengkapnya, lihat model tanggung jawab AWS bersama.

Memigrasi beban kerja dari AWS Data Pipeline ke Step Functions

AWS meluncurkan AWS Data Pipeline layanan pada tahun 2012. Pada saat itu, pelanggan menginginkan layanan yang memungkinkan mereka menggunakan berbagai opsi komputasi untuk memindahkan data antara sumber data yang berbeda. Karena kebutuhan transfer data berubah dari waktu ke waktu, begitu juga solusi untuk kebutuhan tersebut. Anda sekarang memiliki opsi untuk memilih solusi yang paling sesuai dengan kebutuhan bisnis Anda. Misalnya, Anda dapat melakukan hal berikut:

- Gunakan Step Functions untuk mengatur alur kerja antara beberapa. Layanan AWS
- Gunakan Alur Kerja Terkelola Amazon untuk Apache Airflow (Amazon MWAA) untuk mengelola orkestrasi alur kerja untuk Apache Airflow.
- Gunakan AWS Glue untuk menjalankan dan mengatur aplikasi Apache Spark.

Anda dapat memigrasikan kasus penggunaan umum AWS Data Pipeline ke salah satu AWS Glue, Step Functions, atau Amazon MWAA. Opsi yang Anda pilih tergantung pada beban kerja Anda saat ini. AWS Data Pipeline Topik ini menjelaskan cara bermigrasi dari AWS Data Pipeline ke Step Functions.

Topik

- Migrasi beban kerja dari AWS Data Pipeline
- Pemetaan konsep antara Step Functions dan AWS Data Pipeline
- Proyek contoh Step Functions
- Perbandingan harga

Migrasi beban kerja dari AWS Data Pipeline

Step Functions adalah layanan orkestrasi tanpa server tempat Anda membangun alur kerja untuk aplikasi penting bisnis. Dengan Workflow Studio Step Functions, Anda dapat membangun alur kerja dan mengintegrasikannya dengan lebih dari 11.000 tindakan API dari lebih dari 250. Layanan AWS Ini termasuk Layanan AWS seperti AWS Lambda, Amazon EMR, dan Amazon DynamoDB. Anda juga dapat menggunakan Step Functions untuk mengatur pipeline pemrosesan data, menangani

kesalahan, dan bekerja dengan batas pelambatan pada yang mendasarinya. Layanan AWS Anda dapat membuat alur kerja yang memproses dan mempublikasikan model pembelajaran mesin, mengatur layanan mikro, dan menangani alur kerja ekstrak, transformasi, dan muat (ETL). AWS Glue Anda juga dapat membuat alur kerja otomatis yang berjalan lama untuk aplikasi yang memerlukan interaksi manusia.

Step Functions adalah layanan yang dikelola sepenuhnya yang disediakan oleh AWS. Ini berarti AWS mengelola tugas-tugas seperti memelihara infrastruktur, menambal pekerja, dan mengelola pembaruan versi OS untuk Anda.

Jika kasus penggunaan Anda cocok dengan kondisi berikut, sebaiknya Anda bermigrasi dari AWS Data Pipeline ke Step Functions:

- Anda lebih suka layanan orkestrasi alur kerja tanpa server dan sangat tersedia.
- Anda memerlukan solusi yang membebankan biaya pada perincian eksekusi tugas tunggal.
- Beban kerja Anda melibatkan mengatur tugas untuk beberapa tugas lainnya, Layanan AWS seperti Amazon EMR, Lambda, atau DynamoDB. AWS Glue
- Anda memerlukan solusi kode rendah dengan desainer drag-and-drop visual untuk pembuatan alur kerja. Solusi ini seharusnya tidak memerlukan pembelajaran konsep pemrograman yang asing dan kompleks.
- Anda memerlukan layanan yang terintegrasi dengan lebih dari 250 Layanan AWS yang mencakup lebih dari 11.000 tindakan API. Layanan ini juga harus terintegrasi dengan layanan dan aktivitas khusus di luar AWS.

Pemetaan konsep antara Step Functions dan AWS Data Pipeline

AWS Data Pipeline dan Step Functions berbagi beberapa konsep umum. Misalnya, untuk menentukan alur kerja Anda, Anda menggunakan format JSON di keduanya AWS Data Pipeline dan Step Functions. Di Step Functions, Anda menggunakanAmazon States Language, yang merupakan bahasa terstruktur berbasis JSON. Anda menggunakan Amazon States Language (ASL) untuk menentukan alur kerja dan beralih antara representasi tekstual dan visual alur kerja Anda. Format berbasis JSON ini membantu menyederhanakan penyimpanan alur kerja Anda di alat kontrol sumber. Ini juga membantu Anda mengelola beberapa versi alur kerja Anda, mengontrol aksesnya, atau mengotomatiskan orkestrasi mereka dengan metode CI/CD.

Tabel berikut menjelaskan pemetaan antara konsep-konsep utama yang digunakan dalam kedua layanan. Kolom konsep pipa Data di sebelah kiri mencantumkan konsep AWS Data Pipeline,

sedangkan kolom konsep Step Functions di sebelah kanan mencantumkan konsep yang setara di Step Functions.

Konsep pipa data	Konsep Step Functions
Alur	Alur kerja
Definisi pipa	Amazon States Language(ASL)
Aktivitas	Status dan Tugas
Instans	Eksekusi
Upaya	Penangkap dan retrier
Jadwal pipa	• Eksekusi dengan Amazon Scheduler EventBridge • Peristiwa dipicu melalui EventBridge Pipes
Ekspresi dan fungsi pipa	• Fungsi intrinsik • Fungsi Lambda menggunakan integrasi layanan

Proyek contoh Step Functions

Untuk pengenalan Step Functions, lihat video berikut:

Memulai dengan AWS Step Functions untuk orkestrasi layanan

Daftar berikut menguraikan beberapa proyek sampel yang mengimplementasikan kasus AWS Data Pipeline penggunaan paling umum dengan Step Functions. Anda dapat menggunakan proyek contoh ini sebagai referensi untuk bermigrasi dari AWS Data Pipeline ke Step Functions. Anda juga dapat menggunakannya sebagai boilerplate untuk membangun alur kerja Anda sendiri dan mengintegrasikan dengan yang didukung Layanan AWS berdasarkan kasus penggunaan Anda.

- Kelola Tugas Amazon EMR
- Jalankan pekerjaan pemrosesan data di Amazon EMR Tanpa Server
- lowongan kerja Running Hive/Big/Hadoop

- Kueri kumpulan data besar (Amazon Athena, Amazon S3,, AWS Glue Amazon SNS)
- Jalankan alur kerja ETL/ELT menggunakan Amazon Redshift
- Mengatur crawler AWS Glue
- Jalankan skrip shell dengan Step Functions

Untuk mempelajari lebih lanjut tentang Step Functions, lihat topik dan sumber daya berikut:

- Tutorial untuk Step Functions
- Proyek sampel untuk Step Functions
- AWS Step Functions Lokakarya

Perbandingan harga

AWS Data Pipeline dihargai dengan jumlah jaringan pipa dan tingkat penggunaannya. Aktivitas yang dijalankan lebih dari sekali sehari (frekuensi tinggi) dihargai $1 per bulan per aktivitas. Aktivitas yang dijalankan sekali sehari atau kurang (frekuensi rendah) dihargai $0,60 per bulan per aktivitas. Pipa Tidak Aktif dihargai $1 per pipa. Untuk informasi selengkapnya tentang harga, lihat halaman AWS Data Pipeline Harga.

Step Functions memiliki dua jenis alur kerja: Standard dan Express. Setiap jenis alur kerja memiliki model harga yang berbeda. Perbandingan ini didasarkan pada alur kerja Standar karena paling cocok dengan kasus penggunaan umum dari AWS Data Pipeline. Alur kerja standar dihargai $0,025 per 1000 transisi status. Tidak ada biaya untuk mesin negara tidak aktif; Anda hanya membayar untuk apa yang Anda gunakan. Untuk informasi selengkapnya tentang harga, lihat halaman AWS Step Functions Harga.

Pemecahan Masalah

Jika Anda mengalami kesulitan saat menggunakan Step Functions, gunakan sumber daya pemecahan masalah berikut.

Topik

- Pemecahan masalah umum
- Memecahkan masalah integrasi layanan
- Memecahkan masalah aktivitas
- Memecahkan masalah alur kerja ekspres

Pemecahan masalah umum

Saya tidak bisa membuat mesin status.

IAM role yang terkait dengan mesin status mungkin tidak memiliki izin memadai. Periksa izin peran IAM, termasuk untuk tugas integrasi AWS layanan, X-Ray, dan CloudWatch logging. Izin tambahan diperlukan untuk status tugas `.sync`.

Saya tidak dapat menggunakan JsonPath untuk referensi output tugas sebelumnya.

UntukJsonPath, kunci JSON harus diakhiri dengan`.$`. Ini berarti hanya JsonPath dapat digunakan dalam pasangan kunci-nilai. Jika Anda ingin menggunakan tempat JsonPath lain, seperti array, Anda dapat menggunakan fungsi intrinsik. Misalnya, Anda dapat menggunakan sesuatu yang serupa dengan yang berikut ini:

Output tugas A:

```
{
    "sample": "test"
}
```

Tugas B:

```
{
    "JsonPathSample.$": "$.sample"
}
```

> **Tip**
>
> Gunakan simulator aliran data di konsol Step Functions untuk menguji sintaks jalur JSON, untuk lebih memahami bagaimana data dimanipulasi dalam keadaan, dan untuk melihat bagaimana data dilewatkan antar status.

Ada penundaan dalam transisi status.

Untuk alur kerja standar, ada batas jumlah transisi status. Bila Anda melebihi batas transisi status, Step Functions menunda transisi status sampai bucket untuk kuota diisi. Pembatasan batas transisi status dapat dipantau dengan meninjau `ExecutionThrottled` metrik di Metrik eksekusi bagian halaman Metrik. CloudWatch

Ketika saya mulai eksekusi alur kerja standar baru, mereka gagal dengan kesalahan **`ExecutionLimitExceeded`**.

Step Functions memiliki batas 1.000.000 eksekusi terbuka untuk masing-masingAkun AWS. Wilayah AWS Jika Anda melebihi batas ini, Step Functions melempar kesalahan `ExecutionLimitExceeded`. Batas ini tidak berlaku untuk alur kerja Express. Anda dapat menggunakan CloudWatchmatematika Metrik berikut di Panduan CloudWatch Pengguna Amazon untuk memperkirakan jumlah eksekusi terbuka:. `ExecutionsStarted - (ExecutionsSucceeded + ExecutionsTimedOut + ExecutionsFailed + ExecutionsAborted)`

Kegagalan pada satu cabang dalam keadaan paralel menyebabkan seluruh eksekusi gagal.

Ini adalah perilaku yang diharapkan. Untuk menghindari menghadapi kegagalan saat menggunakan keadaan paralel, konfigurasikan Step Functions untuk menangkap kesalahan dilemparkan dari setiap cabang.

Memecahkan masalah integrasi layanan

Pekerjaan saya selesai di layanan hilir, tetapi dalam Step Functions status tugas tetap “Sedang berlangsung” atau penyelesaiannya tertunda.

Untuk pola integrasi `.sync` layanan, Step Functions menggunakan EventBridge aturan, API hilir, atau kombinasi keduanya untuk mendeteksi status pekerjaan hilir. Untuk beberapa layanan, Step Functions tidak membuat EventBridge aturan untuk dipantau. Misalnya, untuk integrasi AWS Glue layanan, alih-alih menggunakan EventBridge aturan, Step Functions membuat `glue:GetJobRun` panggilan. Karena frekuensi panggilan API, ada perbedaan antara penyelesaian tugas downstream dan waktu penyelesaian tugas Step Functions. Step Functions memerlukan izin IAM untuk mengelola EventBridge aturan dan melakukan panggilan ke layanan hilir. Untuk detail selengkapnya tentang bagaimana izin yang tidak memadai pada peran eksekusi Anda dapat memengaruhi penyelesaian tugas, lihatIzin tambahan untuk tugas menggunakan pola Jalankan Tugas.

Saya ingin mengembalikan output JSON dari eksekusi mesin status nest.

Ada dua integrasi layanan sinkron Step Functions untuk Step Functions: `startExecution.sync` dan `startExecution.sync:2`. Keduanya menunggu mesin status nest selesai, tetapi layanan mengembalikan format `Output` yang berbeda. Anda dapat menggunakan `startExecution.sync:2` untuk mengembalikan output JSON di bawah `Output`.

Saya tidak bisa memanggil fungsi Lambda dari akun lain.

Mengakses fungsi Lambda dengan dukungan lintas akun

Jika akses lintas akun AWS sumber daya tersedia di Wilayah Anda, gunakan metode berikut untuk menjalankan fungsi Lambda dari akun lain.

Untuk memanggil sumber daya lintas akun di alur kerja Anda, lakukan hal berikut:

1. Buat peran IAM di akun target yang berisi sumber daya. Peran ini memberikan izin kepada akun sumber, yang berisi mesin status, untuk mengakses sumber daya akun target.
2. Dalam definisi `Task` negara, tentukan peran IAM target yang akan diasumsikan oleh mesin negara sebelum memanggil sumber daya lintas akun.
3. Ubah kebijakan kepercayaan dalam peran IAM target untuk memungkinkan akun sumber untuk mengambil peran ini sementara. Kebijakan kepercayaan harus menyertakan Amazon Resource

Name (ARN) dari mesin status yang ditentukan dalam akun sumber. Juga, tentukan izin yang sesuai dalam peran IAM target untuk memanggil sumber daya. AWS

4. Perbarui peran eksekusi akun sumber untuk menyertakan izin yang diperlukan untuk mengasumsikan peran IAM target.

Sebagai contoh, lihat Tutorial: Mengakses sumber daya lintas akun AWS.

> **Note**
>
> Anda dapat mengkonfigurasi mesin negara Anda untuk mengambil peran IAM untuk mengakses sumber daya dari beberapa. Akun AWS Namun, mesin negara hanya dapat mengasumsikan satu peran IAM pada waktu tertentu.

Untuk contoh definisi `Task` status yang menentukan sumber daya lintas akun, lihat. Contoh bidang Kredensial status tugas

Mengakses fungsi Lambda tanpa dukungan lintas akun

Jika akses lintas akun AWS sumber daya tidak tersedia di Wilayah Anda, gunakan metode berikut untuk memanggil fungsi Lambda dari akun lain.

Di bidang `Resource` status `Task`, gunakan `arn:aws:states:::lambda:invoke` dan lewati `FunctionArn` dalam parameter. IAM role yang terkait dengan mesin status harus memiliki izin yang tepat untuk memanggil fungsi Lambda lintas-akun: `lambda:invokeFunction`.

```
{
   "StartAt":"CallLambda",
   "States":{
      "CallLambda":{
         "Type":"Task",
         "Resource":"arn:aws:states:::lambda:invoke",
         "Parameters":{
            "FunctionName":"arn:aws:lambda:us-west-2:123456789012:function:my-function"
         },
         "End":true
      }
   }
}
```

Saya tidak dapat melihat token tugas diteruskan dari status `.waitForTaskToken`.

Di bidang `Parameters` status `Task`, Anda harus melewati token tugas. Misalnya, Anda dapat menggunakan sesuatu yang mirip dengan kode berikut.

```
{
   "StartAt":"taskToken",
   "States":{
      "taskToken":{
         "Type":"Task",
         "Resource":"arn:aws:states:::lambda:invoke.waitForTaskToken",
         "Parameters":{
            "FunctionName":"get-model-review-decision",
            "Payload":{
               "token.$":"$$.Task.Token"
            },
         },
         "End":true
      }
   }
}
```

> **Note**
>
> Anda dapat mencoba menggunakan `.waitForTaskToken` dengan tindakan API apa pun. Namun, beberapa API tidak memiliki parameter yang sesuai.

Memecahkan masalah aktivitas

Eksekusi mesin negara saya terjebak pada keadaan aktivitas.

Status tugas aktivitas tidak dimulai hingga Anda melakukan polling token tugas dengan menggunakan tindakan GetActivityTaskAPI. Sebagai praktik terbaik, menambahkan tugas tingkat batas waktu untuk menghindari eksekusi yang terjebak. Untuk informasi selengkapnya, lihat Gunakan timeout untuk menghindari eksekusi macet.

Jika mesin negara Anda macet dalam ActivityScheduledacara tersebut, ini menunjukkan bahwa armada pekerja aktivitas Anda memiliki masalah atau underscaled. Anda harus memantau

`ActivityScheduleTime`CloudWatchmetrik dan mengatur alarm saat waktu itu meningkat. Namun, untuk waktu keluar setiap eksekusi mesin negara macet di mana `Activity` negara tidak transisi ke `ActivityStarted` negara, tentukan batas waktu di tingkat mesin negara. Untuk melakukan ini, tentukan `TimeoutSeconds` bidang di awal definisi mesin negara, di luar `States` lapangan.

Waktu pekerja aktivitas saya habis saat menunggu token tugas.

Pekerja menggunakan tindakan GetActivityTaskAPI untuk mengambil tugas dengan ARN aktivitas tertentu yang dijadwalkan untuk dieksekusi oleh mesin status yang sedang berjalan. `GetActivityTask`memulai jajak pendapat panjang, sehingga layanan menahan koneksi HTTP terbuka dan merespons segera setelah tugas tersedia. Waktu maksimum layanan menahan permintaan sebelum merespons adalah 60 detik. Jika tidak ada tugas tersedia dalam waktu 60 detik, jajak pendapat mengembalikan `taskToken` dengan string nol. Untuk menghindari batas waktu ini, konfigurasikan soket sisi klien dengan batas waktu minimal 65 detik dalam AWS SDK atau klien yang Anda gunakan untuk membuat panggilan API.

Memecahkan masalah alur kerja ekspres

Aplikasi saya habis waktunya sebelum menerima tanggapan dari **`StartSyncExecution`** Panggilan API.

Konfigurasikan batas waktu tunggu soket sisi klien di AWS SDK atau klien yang Anda gunakan untuk melakukan panggilan API. Untuk menerima respons, batas waktu harus memiliki nilai yang lebih tinggi dari durasi eksekusi Alur Kerja Ekspres.

Saya tidak dapat melihat sejarah eksekusi untuk memecahkan masalah kegagalan alur kerja ekspres.

Alur Kerja Ekspres tidak merekam riwayat eksekusi diAWS Step Functions. Sebagai gantinya, Anda harus mengaktifkan CloudWatch logging. Setelah pencatatan diaktifkan, Anda dapat menggunakan kueri Wawasan CloudWatch Log untuk meninjau eksekusi Alur Kerja Ekspres. Anda juga dapat melihat riwayat eksekusi untuk eksekusi Alur Kerja Ekspres di konsol Step Functions jika Anda memilih tombol Aktifkan di tab Eksekusi. Untuk informasi selengkapnya, lihat Melihat dan men-debug eksekusi di konsol Step Functions.

Untuk membuat daftar eksekusi berdasarkan durasi:

```
fields ispresent(execution_arn) as exec_arn
```

```
| filter exec_arn
| filter type in ["ExecutionStarted", "ExecutionSucceeded", "ExecutionFailed",
 "ExecutionAborted", "ExecutionTimedOut"]
| stats latest(type) as status,
  tomillis(earliest(event_timestamp)) as UTC_starttime,
  tomillis(latest(event_timestamp)) as UTC_endtime,
  latest(event_timestamp) - earliest(event_timestamp) as duration_in_ms  by
 execution_arn
| sort duration desc
```

Untuk membuat daftar eksekusi yang gagal dan dibatalkan:

```
fields ispresent(execution_arn) as isRes | filter type in ["ExecutionFailed",
 "ExecutionAborted", "ExecutionTimedOut"]
```

Informasi terkait

Berikut ini adalah tabel yang mencantumkan daftar sumber daya terkait yang akan berguna saat Anda bekerja dengan layanan ini.

Sumber Daya	Deskripsi
AWS Step FunctionsReferensi API	Deskripsi tentang tindakan API, parameter, dan tipe data serta daftar kesalahan yang dikembali kan layanan.
AWS Step FunctionsReferensi Baris Perintah	Deskripsi perintah AWS CLI yang dapat Anda gunakan untuk bekerja dengan AWS Step Functions.
Informasi produk untuk Fungsi Langkah	Halaman web utama untuk informasi tentang Step Functions.
Forum Diskusi	Forum berbasis komunitas bagi developer untuk membahas pertanyaan teknis terkait Step Functions dan layanan AWS lainnya.
AWS SupportInformasi	Halaman web utama untuk informasi tentangAWS Support, ataune-on-one, cepat-res pon saluran dukungan untuk membantu Anda membangun dan menjalankan aplikasi pada AWS layanan infrastruktur.

Peluncuran fitur terbaru

Tabel berikut mencantumkan Wilayah di mana fitur Step Functions baru tersedia.

Tanggal peluncuran	Nama fitur	Wilayah yang tersedia
26 November 2023	Memanggil titik akhir HTTPS publik dan menguji status individual	• US East (N. Virginia) – us-east-1 • US West (Oregon) – us-west-2 • US East (Ohio) – us-east-2 • Europe (Ireland) – eu-west-1 • Europe (Frankfurt) – eu-central-1 • Europe (Stockholm) – (eu-north-1) • Asia Pacific (Sydney) – ap-southeast-2 • Asia Pacific (Tokyo) – ap-northeast-1 • Asia Pacific (Singapore) – ap-southeast-1
15 November 2023	Redriveeksekusi	Untuk daftar lengkap Wilayah AWS di mana fitur ini tersedia, lihat opsi di daftar dropdown Wilayah pada halaman berjudul Layanan AWS berdasarkan Wilayah.
12 Oktober 2023	Integrasi yang dioptimalkan untuk Amazon EMR Serverless	Untuk daftar lengkap Wilayah AWS di mana fitur ini tersedia, lihat opsi di daftar dropdown Wilayah pada halaman

Tanggal peluncuran	Nama fitur	Wilayah yang tersedia
		berjudul Layanan AWS berdasarkan Wilayah.
September 07, 2023	Penanganan kesalahan yang ditingkatkan	Untuk daftar lengkap Wilayah AWS di mana fitur ini tersedia, lihat opsi di daftar dropdown Wilayah pada halaman berjudul Layanan AWS berdasarkan Wilayah.
31 Agustus 2023	Penyempurnaan Workflow Studio untuk pengalaman penulisan yang efisien	Untuk daftar lengkap Wilayah AWS di mana fitur ini tersedia, lihat opsi di daftar dropdown Wilayah pada halaman berjudul Layanan AWS berdasarkan Wilayah.
Juni 22, 2023	Versi dan alias	Untuk daftar lengkap Wilayah AWS di mana fitur ini tersedia, lihat opsi di daftar dropdown Wilayah pada halaman berjudul Layanan AWS berdasarkan Wilayah.
Juni 16, 2023	Integrasi AWS SDK baru	Untuk daftar lengkap Wilayah AWS di mana fitur ini tersedia, lihat opsi di daftar dropdown Wilayah pada halaman berjudul Layanan AWS berdasarkan Wilayah.

Tanggal peluncuran	Nama fitur	Wilayah yang tersedia
Desember 01, 2022	Mengatur alur kerja paralel skala besar untuk pemrosesan data dengan status Peta Terdistribusi	Untuk daftar lengkap Wilayah AWS di mana fitur ini tersedia, lihat opsi di daftar dropdown Wilayah pada halaman berjudul Layanan AWS berdasarkan Wilayah.

Riwayat dokumen

Bagian ini mencantumkan perubahan besar pada Panduan Developer AWS Step Functions .

Perubahan	Deskripsi	Tanggal diubah
Fitur baru	Step Functions menambahkan integrasi yang dioptimalkan untuk AWS Elemental MediaConvert AWS Elemental MediaConvert menyediakan transcoding file video dan audio tingkat siaran, yang dapat diotomati skan oleh pelanggan dengan kode agar sesuai dengan alur kerja media mereka. Dengan integrasi yang dioptimal kan untuk AWS Step Functions in MediaConvert, sekarang dimungkinkan untuk mengatur menggunakan alat visual kode rendah Workflow Studio. Untuk mempelajari selengkapnya, lihat dokumentasi untuk Mengelola AWS Elemental MediaConvert dengan Step Functions.	April 12, 2024
Pembaruan	AWS pembaruan kebijakan terkelola - Memperbarui ke kebijakan yang ada Menambahkan informasi tentang izin hanya-baca baru untuk tag, peta terdistribusi, serta versi dan alias. Untuk mempelajari selengkapnya, lihat AWS kebijakan terkelola untuk AWS Step Functions.	April 02, 2024
Pembaruan	Step Functions menambahkan dukungan untuk metrik Open Workflow Dengan metrik alur kerja terbuka, Anda sekarang memiliki visibilitas tingkat akun ke dalam jumlah alur kerja standar yang sedang berlangsung serta batas alur kerja terbuka Anda. Anda dapat mengelola beban kerja di semua alur kerja, terlepas dari bagaimana mereka dimulai, untuk memastikan kelancaran operasi alur kerja. Anda dapat mengatur CloudWatch alarm untuk memantau alur kerja	Februari 29, 2024

Perubahan	Deskripsi	Tanggal diubah
	dan secara proaktif menerima peringatan saat mendekati batas. Setelah diperingatkan, Anda dapat mengelola alur kerja secara efektif dengan mengambil tindakan seperti menghentikan alur kerja tertentu atau meminta peningkatan batas. Metrik alur kerja terbuka tersedia untuk digunakan dalam CloudWatch alur kerja standar tanpa konfigurasi tambahan yang diperlukan. Untuk mempelajari selengkapnya, lihat Metrik eksekusi.	
Pembaruan	Penambahan dan pembaruan integrasi layanan. Untuk daftar integrasi AWS SDK baru dan yang diperbarui, lihat. Ubah log untuk integrasi AWS SDK yang didukung Untuk daftar lengkap layanan, lihatIntegrasi layanan AWS SDK yang didukung.	Januari 18, 2024
Fitur baru	Gunakan Workflow Studio Application Composer untuk membangun alur kerja tanpa server menggunakan templat. AWS CloudFormation Untuk informasi selengkapnya, lihat Menggunakan Workflow Studio di Application Composer.	27 November 2023
Fitur baru	Step Functionssekarang memungkinkan Anda langsung memanggil titik akhir HTTPS publik dan menguji status individual menggunakan API Status Uji baru. Lihat informasi yang lebih lengkap di: • Panggil API pihak ketiga • Menggunakan TestState API untuk menguji status	26 November 2023

Perubahan	Deskripsi	Tanggal diubah
Fitur baru	Step Functions sekarang terintegrasi dengan Amazon Bedrock. Untuk informasi selengkapnya, lihat topik berikut: • Panggilan Amazon Bedrock dengan Step Functions • IAMizin untuk Amazon Bedrock • Lakukan AI prompt-chaining dengan Amazon Bedrock • Menggunakan AWS Step Functions dengan layanan lain	26 November 2023
Fitur baru	Step Functions sekarang memungkinkan Anda menjalank an eksekusi redrive workflow tipe Standard dari titik kegagalannya. Lihat informasi yang lebih lengkap di Redrivingeksekusi dan RedrivingPeta Berjalan.	15 November 2023
Pembaruan khusus dokumentasi	Menerbitkan topik baru yang menjelaskan cara menjalank an mesin status sesuai jadwal menggunakanAmazon EventBridge Scheduler. Untuk informasi selengkapnya, lihat Menggunakan Amazon EventBridge Scheduler dengan AWS Step Functions.	16 Oktober 2023
Fitur baru	Step Functions sekarang terintegrasi dengan Amazon EMR Serverless. Untuk informasi selengkapnya, lihat topik berikut: • Panggilan Amazon EMR Serverless dengan Step Functions • Jalankan EMR Serverless pekerjaan • Integrasi yang dioptimalkan untuk Step Functions • Menggunakan AWS Step Functions dengan layanan lain	12 Oktober 2023
Pembaruan khusus dokumentasi	Menambahkan informasi tentang menjalankan mesin negara sesuai jadwal menggunakanAmazon EventBridge Scheduler. Untuk informasi selengkapnya, lihat Menggunak an EventBridge Scheduler.	Oktober 05, 2023

Perubahan	Deskripsi	Tanggal diubah
Perbarui	Menata ulang dan memperbarui topik status Peta Terdistribusi untuk kejelasan, singkatnya, dan membuat peta perjalanan yang jelas bagi pengguna baru. Untuk informasi selengkapnya, lihat Menggunakan status Peta dalam mode Terdistribusi untuk mengatur beban kerja paralel skala besar.	Oktober 6, 2023
Perbaikan	Sampel kode tetap dalam tutorial untuk menggunakan AWS CDK v2. Untuk informasi selengkapnya, lihat Membuat mesin Lambda negara untuk Step Functions digunakan AWS CDK.	September 19, 2023
Perbarui	Menambahkan informasi tentang kemampuan penanganan kesalahan yang ditingkatkan yang telah diperkenalkan Step Functions untuk mengidentifikasi kesalahan dengan jelas dan mengimplementasikan percobaan ulang dengan kontrol yang lebih besar. Lihat informasi yang lebih lengkap di Gagal dan Mencoba kembali setelah kesalahan.	September 07, 2023
Perbarui	Step Functions telah menambahkan penyempurnaan ke Workflow Studio untuk merampingkan pengalaman penulisan alur kerja. Untuk informasi selengkapnya, lihat AWS Step Functions Studio Alur Kerja.	31 Agustus 2023
Pembaruan khusus dokumentasi	Menambahkan informasi sekitar dua kali jumlah metrik aktual yang dilaporkan untuk `ExecutionsStarted` metrik. Untuk informasi selengkapnya, lihat Metrik yang melaporkan hitungan.	25 Juli 2023
Pembaruan khusus dokumentasi	Step Functions telah menambahkan dua proyek sampel baru yang menunjukkan kasus penggunaan umum berikut untuk status Peta Terdistribusi: • Memproses file CSV • Memproses data dalam bucket Amazon S3	Juli 17, 2023

Perubahan	Deskripsi	Tanggal diubah
Pembaruan khusus dokumentasi	Menerbitkan topik baru tentang penerapan mesin status menggunakan Terraform. Untuk informasi selengkapnya, lihat Menyebarkan mesin status menggunakan Terraform.	5 Juli 2023
Pembaruan khusus dokumentasi	Memperbarui prosedur berikut untuk mencocokkan perubahan pada EventBridge antarmuka Amazon. • Merutekan acara Step Functions ke EventBridge • Memulai Eksekusi Mesin Status dalam Respons terhadap Peristiwa Amazon S3	26 Juni 2023
Fitur baru	Step Functions sekarang menyediakan kemampuan untuk membuat beberapa versi mesin status dan alias untuk meningkatkan ketahanan saat menerapkan alur kerja tanpa server. Untuk informasi selengkapnya, lihat Mengelola penerapan berkelanjutan dengan versi dan alias.	22 Juni 2023
Pembaruan khusus dokumentasi	Memperbaiki deskripsi `TimeoutSeconds` dan `HeartbeatSeconds` bidang untuk menggambarkan bagaimana mereka berbeda satu sama lain. Untuk informasi selengkapnya, lihat Bidang status tugas.	22 Juni 2023
Pembaruan khusus dokumentasi	Menerbitkan bagian baru yang menjelaskan cara meratakan array array yang biasanya dikembalikan sebagai hasil untuk status Paralel dan Peta. Untuk informasi selengkapnya, lihat Meratakan array array.	20 Juni 2023
Perbarui	Step Functions telah memperluas dukungan untuk integrasi AWS SDK dengan menambahkan tujuh Layanan AWS dan 468 tindakan API baru. Lihat informasi yang lebih lengkap di Integrasi layanan AWS SDK yang didukung dan Ubah log untuk integrasi AWS SDK yang didukung.	Juni 16, 2023

Perubahan	Deskripsi	Tanggal diubah
Pembaruan khusus dokumentasi	Menerbitkan topik baru yang mencantumkan fitur Step Functions yang baru diluncurkan tersedia. Wilayah AWS Untuk informasi selengkapnya, lihat Peluncuran fitur terbaru.	Juni 16, 2023
Pembaruan khusus dokumentasi	Step Functions sekarang mencakup bagian tentang Notifikasi Pengguna AWS, Layanan AWS yang bertindak sebagai lokasi pusat untuk AWS notifikasi Anda di AWS Management Console. Untuk informasi selengkapnya, lihat Menggunakan Notifikasi Pengguna AWS dengan Step Functions.	4 Mei 2023
Pembaruan khusus dokumentasi	Menambahkan bagian baru yang menjelaskan tentang izin yang diperlukan untuk menulis hasil eksekusi alur kerja turunan ke bucket Amazon S3 yang dienkripsi dengan kunci. AWS Key Management Service (AWS KMS) Untuk informasi selengkapnya, lihat Izin IAM untuk bucket AWS KMS key Amazon S3 terenkripsi.	April 25, 2023
Pembaruan khusus dokumentasi	Menambahkan topik baru yang menjelaskan tentang fitur simulator aliran Data. Untuk informasi selengkapnya, lihat Simulator aliran data.	April 14, 2023
Pembaruan kuota	Menambahkan informasi tentang kuota default 1000 untuk membuka Map Runs di setiap akun. Untuk informasi selengkapnya, lihat Kuota yang terkait dengan akun.	April 05, 2023
Pembaruan khusus dokumentasi	Menambahkan topik yang menjelaskan kapan harus memigrasikan AWS Data Pipeline beban kerja ke Step Functions. Topik ini juga menyediakan daftar contoh yang menjelaskan cara melakukan migrasi. Untuk informasi selengkapnya, lihat Memigrasi beban kerja dari AWS Data Pipeline ke Step Functions.	30 Maret 2023

Perubahan	Deskripsi	Tanggal diubah
Pembaruan khusus dokumentasi	Menambahkan Catatan tentang tidak tersedianya penelusuran X-Ray untuk status Peta Terdistribusi. Untuk informasi selengkapnya, lihat AWS X-Ray dan Step Functions.	21 Maret 2023
Pembaruan khusus dokumentasi	Menambahkan informasi tentang bagaimana Step Functions menangani otorisasi berbasis tag. Lihat informasi yang lebih lengkap di Penandaan di Step Functions dan Kebijakan berbasis tanda.	15 Maret 2023
Pembaruan khusus dokumentasi	Menambahkan informasi tentang bagaimana Step Functions mem-parsing file CSV yang digunakan sebagai input dalam status Peta Terdistribusi. Untuk informasi selengkapnya, lihat File CSV dalam ember Amazon S3.	14 Maret 2023
Pembaruan khusus dokumentasi	Menambahkan informasi tentang cara Step Functions menangani pemanggilan lintas akun untuk pola Run a Job (.sync). Untuk informasi selengkapnya, lihat Menjalankan Job (.sync).	01 Maret, 2023
Pembaruan khusus dokumentasi	Kurangi periode retensi riwayat eksekusi alur kerja Anda yang telah selesai dari 90 hari menjadi 30 hari. Untuk informasi selengkapnya tentang menyesuaikan periode retensi, lihat Jaminan eksekusi danKuota yang berkaitan dengan eksekusi mesin status.	21 Februari 2023
Perbarui	Step Functions telah memperluas dukungan untuk integrasi AWS SDK dengan menambahkan 35 AWS layanan dan 1100 tindakan API baru. Lihat informasi yang lebih lengkap di Integrasi layanan AWS SDK yang didukung dan Ubah log untuk integrasi AWS SDK yang didukung.	17 Februari 2023

Perubahan	Deskripsi	Tanggal diubah
Pembaruan khusus dokumentasi	Menerbitkan seri tutorial Getting Started yang memandu Anda melalui proses pembuatan alur kerja untuk aplikasi kartu kredit menggunakan Step Functions. Untuk informasi selengkapnya, lihat Memulai dengan AWS Step Functions.	30 Desember 2022
Fitur baru	Step Functions menambahkan dukungan untuk mengatur alur kerja paralel skala besar untuk pemrosesan data menggunakan mode Distributed baru untuk status. `Map` Untuk informasi selengkapnya, lihat Menggunakan status Peta dalam mode Terdistribusi untuk mengatur beban kerja paralel skala besar.	Desember 01, 2022
Fitur baru	Step Functions sekarang mendukung akses ke AWS sumber daya lintas akun yang dikonfigurasi di akun lain. Untuk informasi selengkapnya, silakan lihat • Mengakses sumber daya di tempat lain Akun AWS di alur kerja Anda • Tutorial: Mengakses sumber daya lintas akun AWS • Task state	18 November 2022
Perbarui	Step Functions sekarang menyediakan pengalaman konsol baru untuk melihat dan men-debug eksekusi alur kerja Express. Untuk informasi selengkapnya, lihat: • Eksekusi Alur Kerja Standar dan Ekspres di konsol • Melihat dan men-debug eksekusi di konsol Step Functions	18 Oktober 2022

Perubahan	Deskripsi	Tanggal diubah
Perbarui	Menambahkan dukungan untuk menentukan `ExecutionRoleArn` parameter secara opsional saat menggunakan `addStep` dan `addStep.sync` API untuk integrasi layanan yang dioptimalkan Amazon EMR. Untuk informasi selengkapnya, lihat Panggil Amazon EMR dengan Step Functions.	September 20, 2022
Pembaruan khusus dokumentasi	Menambahkan topik baru yang memberikan rekomendasi tentang mengoptimalkan biaya sambil membangun alur kerja tanpa server menggunakan Step Functions. Untuk informasi selengkapnya, lihat Optimalisasi biaya menggunakan Alur Kerja Ekspres.	15 September 2022

Perubahan	Deskripsi	Tanggal diubah
Perbarui	Step Functions menambahkan dukungan untuk 14 fungsi intrinsik baru untuk melakukan tugas pemrosesan data, seperti manipulasi array, pengkodean dan decoding data, perhitungan hash, manipulasi data JSON, operasi fungsi matematika, dan pembuatan pengenal unik. Pembaruan khusus dokumentasi: Mengelompokkan semua fungsi intrinsik yang ada dan yang baru diperkenalkan ke dalam kategori berikut berdasarkan jenis tugas pemrosesan data yang mereka bantu Anda lakukan: • Intrinsik untuk array • Intrinsik untuk pengkodean dan decoding data • Intrinsik untuk perhitungan hash • Intrinsik untuk manipulasi data JSON • Intrinsik untuk operasi Matematika • Intrinsik untuk operasi String • Intrinsik untuk pembuatan pengenal unik • Intrinsik untuk operasi generik Untuk informasi selengkapnya, lihat Fungsi intrinsik.	31 Agustus 2022
Perbarui	Step Functions telah memperluas dukungan untuk integrasi AWS SDK dengan menambahkan tiga AWS layanan lagi —AWS Billing Conductor,Amazon GameSparks, dan. Amazon Pinpoint SMS and Voice V2 Untuk informasi selengkapnya, lihat Ubah log untuk integrasi AWS SDK yang didukung.	26 Juli 2022

Perubahan	Deskripsi	Tanggal diubah
Pembaruan khusus dokumentasi	Menambahkan topik baru untuk menyertakan ringkasan semua pembaruan yang dibuat untuk integrasi AWS SDK yang didukung oleh Step Functions. Lihat informasi yang lebih lengkap di Ubah log untuk integrasi AWS SDK yang didukung	26 Juli 2022
Pembaruan khusus dokumentasi	AWS Step Functions Panduan Pengembang sekarang mencakup detail tentang metrik eksekusi yang dipancark an khusus untuk Alur Kerja Ekspres. Untuk informasi selengkapnya, lihat Metrik eksekusi untuk Alur Kerja Ekspres.	Juni 09, 2022

Perubahan	Deskripsi	Tanggal diubah
Perbarui	Penyempurnaan konsol Step Functions Konsol sekarang menampilkan halaman Detail Eksekusi yang didesain ulang yang mencakup penyempurnaan berikut: • Kemampuan untuk mengidentifikasi alasan eksekusi gagal secara sekilas. • Dua mode visualisasi baru untuk mesin status Anda - Tampilan tabel dan tampilan Acara. Tampilan ini juga memberi Anda kemampuan untuk menerapkan filter untuk hanya melihat informasi yang menarik. Selain itu, Anda dapat mengurutkan konten tampilan Acara berdasarkan stempel waktu acara. • Beralih di antara iterasi `Map` status yang berbeda dalam mode tampilan Grafik menggunakan daftar tarik-turun atau dalam tampilan pohon mode tampilan tabel untuk status. `Map` • Lihat informasi mendalam tentang setiap status dalam alur kerja, termasuk jalur transfer data input dan output lengkap dan coba lagi untuk `Task` atau status. `Parallel` • Berbagai penyempurnaan termasuk opsi untuk menyalin eksekusi mesin status Nama Sumber Daya Amazon, melihat jumlah total transisi mesin status, dan mengekspor detail eksekusi dalam format JSON. Pembaruan khusus dokumentasi Menambahkan topik baru untuk menjelaskan berbagai jenis informasi yang ditampilkan di halaman Rincian Eksekusi. Juga, menambahkan tutorial untuk menunjukkan	Mei 09, 2022

Perubahan	Deskripsi	Tanggal diubah
	bagaimana memeriksa informasi ini. Lihat informasi yang lebih lengkap di: • Melihat dan men-debug eksekusi di konsol Step Functions • Tutorial: Memeriksa eksekusi mesin status menggunakan konsol Step Functions	
Perbarui	Step Functions sekarang menyediakan solusi untuk mencegah masalah keamanan deputi yang membingun gkan, yang muncul ketika entitas (layanan atau akun) dipaksa oleh entitas yang berbeda untuk melakukan tindakan. Lihat informasi yang lebih lengkap di: • Mencegah masalah wakil lintas layanan yang membingungkan	02 Mei 2022

Perubahan	Deskripsi	Tanggal diubah

Perbarui	• Step Functions telah memperluas dukungan untuk integrasi AWS SDK dengan menambahkan 21 layanan lainnya AWS . Untuk informasi lebih lanjut, lihat:Integrasi layanan AWS SDK yang didukung.
	• Pembaruan khusus dokumentasi:
	  • Menambahkan daftar semua awalan pengecualian yang ada dalam pengecualian yang dihasilkan saat Anda salah melakukan AWS integrasi layanan SDK dengan Step Functions. Untuk informasi lebih lanjut, lihat:Integrasi layanan AWS SDK yang didukung.
	  • Menambahkan daftar semua tindakan API yang tidak didukung untuk integrasi AWS SDK yang didukung. Untuk informasi lebih lanjut, lihat:Tindakan API yang tidak didukung untuk layanan yang didukung.
	  • Menambahkan daftar semua integrasi AWS SDK yang didukung yang sekarang tidak digunakan lagi. Untuk informasi lebih lanjut, lihat:Integrasi layanan SDK yang tidak digunakan AWS lagi.
		19 April 2022

Fitur baru	Step Functions Local sekarang mendukung integrasi AWS SDK dan mengejek integrasi layanan. Lihat informasi yang lebih lengkap di: • Menggunakan Integrasi Layanan Mocked	28 Januari 2022
Fitur baru	AWS Step Functions sekarang mendukung pembuatan API REST Amazon API Gateway dengan mesin status ekspres sinkron sebagai integrasi backend menggunakan. AWS Cloud Development Kit (AWS CDK) Lihat informasi yang lebih lengkap di: • Membuat API REST API Gateway dengan Mesin Status Ekspres Sinkron Menggunakan AWS CDK	Desember 10, 2021

Perubahan	Deskripsi	Tanggal diubah
Perbarui	Step Functions telah menambahkan tiga proyek sampel baru yang menunjukkan integrasi Step Functions dan konsol Amazon Athena yang ditingkatkan. Lihat informasi yang lebih lengkap di: • Jalankan beberapa kueri (Amazon Athena, Amazon SNS) • Kueri kumpulan data besar (Amazon Athena, Amazon S3,, AWS Glue Amazon SNS) • Tetap perbarui data (Amazon Athena, Amazon S3,) AWS Glue	22 November 2021
Fitur baru	Step Functions telah menambahkan dukungan endpoint Amazon VPC untuk Alur Kerja Synchronous Express. Lihat informasi yang lebih lengkap di: • Amazon VPC Endpoints untuk Step Functions	15 November 2021
Perbarui	AWS Step Functions telah menambahkan tiga proyek sampel baru yang menunjukkan cara menggunakan AWS Batch integrasi Step Functions. Lihat informasi yang lebih lengkap di: • Menggemari AWS Batch pekerjaan • AWS Batch dengan Lambda • Gunakan Step Functions dan AWS Batch dengan penanganan kesalahan	Oktober 14, 2021

Perubahan	Deskripsi	Tanggal diubah
Fitur baru	AWS Step Functions telah menambahkan integrasi AWS SDK, memungkinkan Anda menggunakan tindakan API untuk semua lebih dari dua ratus AWS layanan. Lihat informasi yang lebih lengkap di: • AWS Integrasi layanan SDK • Kumpulkan info bucket Amazon S3 menggunakan integrasi layanan AWS SDK	30 September 2021
Fitur baru	AWS Step Functions telah menambahkan desainer alur kerja visual, AWS Step Functions Workflow Studio. Lihat informasi yang lebih lengkap di: • AWS Step Functions Studio Alur Kerja • Belajar menggunakan AWS Step Functions Workflow Studio	17 Juni 2021
Perbarui	AWS Step Functions telah menambahkan empat API baru`StartBuildBatch` ,`StopBuildBatch` ,, `RetryBuildBatch` dan`DeleteBuildBatch` , ke CodeBuild integrasi. Lihat informasi yang lebih lengkap di: • Panggilan AWS CodeBuild dengan Step Functions	4 Juni 2021
Fitur baru	AWS Step Functions sekarang terintegrasi dengan Amazon EventBridge. Lihat informasi yang lebih lengkap di: • Panggilan EventBridge dengan Step Functions • Kebijakan IAM untuk Step Functions dan Kebijakan IAM untuk Amazon EventBridge • Proyek sampel yang menunjukkan cara Kirim acara khusus ke EventBridge	14 Mei 2021

Perubahan	Deskripsi	Tanggal diubah
Pembaruan	AWS Step Functions telah menambahkan proyek sampel baru yang menunjukkan cara menggunakan Step Functions dan Amazon Redshift Data API untuk menjalankan alur kerja ETL/ELT. Lihat informasi yang lebih lengkap di: • Jalankan alur kerja ETL/ELT menggunakan Amazon Redshift (Lambda, API Data Amazon Redshift)	16 April 2021
Fitur baru	AWS Step Functions memiliki simulator aliran data baru di konsol. Lihat informasi yang lebih lengkap di: • Konsol Step Functions	8 April 2021
Fitur baru	AWS Step Functions sekarang terintegrasi dengan Amazon EMR di EKS. Lihat informasi yang lebih lengkap di: • Hubungi Amazon EMR di EKS dengan AWS Step Functions	29 Maret 2021
Perbarui	Dukungan YAML untuk ketetapan mesin status telah ditambahkan ke AWS Toolkit for Visual Studio Code dan AWS CloudFormation. Lihat informasi yang lebih lengkap di: • Dukungan format ketetapan • AWS Toolkit for Visual Studio Code	4 Maret 2021
Fitur baru	AWS Step Functions sekarang terintegrasi dengan AWS Glue DataBrew. Lihat informasi yang lebih lengkap di: • Kelola AWS Glue DataBrew Pekerjaan dengan Step Functions • Apa itu AWS Glue DataBrew? dalam panduan DataBrew pengembang.	6 Januari 2021

Perubahan	Deskripsi	Tanggal diubah
Fitur baru	AWS Step Functions Alur Kerja Ekspres Sinkron sekarang tersedia, memberi Anda cara mudah untuk mengatur layanan mikro. Lihat informasi yang lebih lengkap di: • Alur kerja Ekspres Sinkron dan Tidak Sinkron • Proyek sampel yang menunjukkan cara Memanggil Alur Kerja Express Sinkron • Dokumentasi StartSyncExecutionAPI.	24 November 2020
Fitur baru	AWS Step Functions sekarang terintegrasi dengan Amazon API Gateway. Lihat informasi yang lebih lengkap di: • Panggil API Gateway dengan Step Functions • Kebijakan IAM untuk Step Functions dan Kebijakan IAM untuk Amazon API Gateway • Proyek sampel yang menunjukkan cara Buat panggilan ke API Gateway	17 November 2020
Fitur baru	AWS Step Functions sekarang terintegrasi dengan Amazon Elastic Kubernetes Service. Lihat informasi yang lebih lengkap di: • Panggil Amazon EKS dengan Step Functions • Kebijakan IAM untuk Step Functions dan Kebijakan IAM untuk Amazon EKS • Proyek sampel yang menunjukkan cara Mengelola klaster Amazon EKS	16 November 2020

Perubahan	Deskripsi	Tanggal diubah
Fitur baru	AWS Step Functions sekarang terintegrasi dengan Amazon Athena. Lihat informasi yang lebih lengkap di: • Panggil Athena dengan Step Functions • Kebijakan IAM untuk Step Functions dan Kebijakan IAM untuk Amazon Athena • Proyek sampel yang menunjukkan cara Mulai kueri Athena	22 Oktober 2020
Fitur baru	AWS Step Functions sekarang mendukung penelusur an end-to-end alur kerja dengan AWS X-Ray, memberi Anda visibilitas penuh di seluruh eksekusi mesin status dan membuatnya lebih mudah untuk menganalisis dan men-debug aplikasi terdistribusi Anda. Lihat informasi yang lebih lengkap di: • AWS X-Ray dan Step Functions • Kebijakan IAM untuk Step Functions dan Kebijakan IAM untuk AWS X-Ray • AWS Step Functions Referensi API • TracingConfiguration	14 September 2020

Perubahan	Deskripsi	Tanggal diubah

Perbarui	AWS Step Functions sekarang mendukung ukuran payload hingga 256 KB data sebagai string yang dikodekan UTF-8. Hal ini memungkinkan Anda memproses muatan yang lebih besar di alur kerja Standar dan Ekspres.
	
	Mesin status Anda yang ada tidak perlu diubah untuk menggunakan muatan yang lebih besar. Namun, Anda akan perlu untuk memperbarui ke versi terbaru SDK Step Functions dan Runner Lokal untuk menggunakan API terbaru. Untuk informasi selengkapnya, lihat:
	
	• Kuota
	• the section called “Gunakan ARN Amazon S3 bukan meneruskan muatan besar”
	• `States.DataLimitExceeded`
	• the section called “CloudWatchLog muatan”
	• the section called “EventBridge muatan”
	• AWS Step Functions Referensi API
	
	  • CloudWatchEventsExecutionDataDetails
	  • HistoryEventExecutionDataDetails
	  • GetExecutionHistory
	  • ActivityScheduledEventDetails
	  • ActivitySucceededEventDetails
	  • CloudWatchEventsExecutionDataDetails
	  • ExecutionSucceededEventDetails
	  • LambdaFunctionScheduledEventDetails
	  • ExecutionSucceededEventDetails
	  • StateEnteredEventDetails
	  • StateExitedEventDetails
	  • TaskSubmittedEventDetails
		3 September 2020

Perubahan	Deskripsi	Tanggal diubah
	• TaskSucceededEventDetails	

Perubahan	Deskripsi	Tanggal diubah

Pembaruan	Amazon States Language telah diperbarui sebagai berikut:
	• Aturan Pilihan telah ditambahkan
	  • Operator perbandingan null, `IsNull`. Tes `IsNull` terhadap nilai null JSON, dan dapat digunakan untuk mendeteksi jika output dari status sebelumnya null atau tidak.
	  • Empat operator baru lainnya telah ditambahkan, IsBoolean, IsNumeric, IsString dan IsTimestamp.
	  • Sebuah tes untuk eksistensi atau non-eksistensi dari sebuah bidang menggunakan operator `IsPresent` . `IsPresent` dapat digunakan untuk mencegah kesalahan `States.Runtime` ketika ada upaya untuk mengakses kunci yang tidak ada.
	  • Pencocokan pola wildcard untuk mendukung perbandingan string terhadap pola dengan satu atau beberapa wildcard.
	  • Perbandingan antara dua variabel untuk operator perbandingan yang didukung.
	• Timeout dan nilai waktu dalam status `Task` sekarang dapat disediakan secara dinamis dari input status bukan nilai tetap menggunakan bidang `TimeoutSeconds Path` dan `HeartbeatSecondsPath` . Lihat status Tugas untuk informasi lebih selengkapnya.
	• Bidang ResultSelector baru menyediakan cara untuk memanipulasi hasil status sebelum `ResultPath` diterapkan. Bidang `ResultSelector` adalah bidang opsional dalam status Map, Paralel, dan Tugas.
	• Fungsi intrinsik telah ditambahkan untuk memungkinkan operasi dasar tanpa status `Task`. Fungsi intrinsik dapat digunakan dalam bidang `Parameters` dan `ResultSelector` .
		13 Agustus 2020

Perubahan	Deskripsi	Tanggal diubah
Perbarui	AWS Step Functions sekarang mendukung panggilan SageMaker `CreateProcessingJob` API Amazon. Lihat informasi yang lebih lengkap di: • Kelola SageMaker dengan Step Functions • Praproses data dan latih model machine learning, proyek sampel yang menunjukkan `CreateProcessingJob` .	4 Agustus 2020
Fitur baru	AWS Step Functions sekarang didukung oleh AWS Serverless Application Model, sehingga lebih mudah untuk mengintegrasikan orkestrasi alur kerja ke dalam aplikasi tanpa server Anda. Lihat informasi yang lebih lengkap di: • AWS Step Functions dan AWS SAM • AWS::Serverless::StateMachine • AWS SAM Template Kebijakan	27 Mei 2020
Fitur baru	AWS Step Functions telah memperkenalkan pemanggil an sinkron baru untuk eksekusi Step Functions yang bersarang. Invokasi baru, `arn:aws:states:::s tates:startExecution.sync:2` , mengembal ikan objek JSON. Invokasi asli, `arn:aws:states:::s tates:startExecution.sync` , tetap didukung, dan mengembalikan string JSON-escaped. Untuk informasi selengkapnya, lihat: • Kelola AWS Step Functions Eksekusi sebagai Layanan Terpadu	19 Mei 2020

Perubahan	Deskripsi	Tanggal diubah
Fitur baru	AWS Step Functions sekarang terintegrasi dengan AWS CodeBuild. Lihat informasi yang lebih lengkap di: • Menggunakan AWS Step Functions dengan layanan lain • Panggilan AWS CodeBuild dengan Step Functions • Integrasi yang dioptimalkan untuk Step Functions	5 Mei 2020
Fitur baru	Step Functions sekarang didukung di AWS Toolkit for Visual Studio Code, sehingga lebih mudah untuk membuat dan memvisualisasikan alur kerja berbasis mesin status tanpa meninggalkan editor kode Anda.	31 Maret 2020
Perbarui	Anda sekarang dapat mengonfigurasi logging ke Amazon CloudWatch Logs untuk alur kerja Standar. Lihat informasi yang lebih lengkap di: • Logging menggunakanCloudWatchLog	25 Februari 2020
Fitur baru	AWS Step Functions Sekarang dapat diakses tanpa memerlukan alamat IP publik, langsung dari Amazon Virtual Private Cloud (VPC). Lihat informasi yang lebih lengkap di: • Amazon VPC Endpoints untuk Step Functions	23 Desember 2019

Perubahan	Deskripsi	Tanggal diubah

Fitur baru	Alur Kerja Ekspres adalah tipe alur kerja baru, cocok untuk beban kerja pemrosesan peristiwa bervolume tinggi seperti penyerapan data IoT, pemrosesan dan transformasi data streaming, dan backend aplikasi seluler.
	
	Untuk informasi lebih lanjut, tinjau topik baru dan pembaruan berikut ini.
	
	• Alur Kerja Standar vs Ekspres
	  • Jaminan eksekusi
	• Menggunakan AWS Step Functions dengan layanan lain
	  • Integrasi yang dioptimalkan untuk Step Functions
	• Proses Pesan Bervolume Tinggi dari Amazon SQS (Alur Kerja Express)
	• Contoh Checkpointing selektif (Alur kerja Express)
	• Kuota
	  • Kuota
	• Logging menggunakanCloudWatchLog
	• AWS Step Functions Referensi API
	  • CreateStateMachine
	  • UpdateStateMachine
	  • DescribeStateMachine
	  • DescribeStateMachineForExecution
	  • StopExecution
	  • DescribeExecution
	  • GetExecutionHistory
	  • ListExecutions
	  • ListStateMachines
	  • StartExecution
	  • CloudWatchLogsLogGroup
		3 Desember 2019

Perubahan	Deskripsi	Tanggal diubah
	• LogDestination • LoggingConfiguration	
Fitur baru	AWS Step Functions sekarang terintegrasi dengan Amazon EMR. Lihat informasi yang lebih lengkap di: • Menggunakan AWS Step Functions dengan layanan lain • Panggil Amazon EMR dengan Step Functions • Integrasi yang dioptimalkan untuk Step Functions	19 November 2019

Perbarui	AWS Step Functions telah merilis AWS Step Functions Data Science SDK. Untuk informasi selengkapnya, lihat hal berikut.
	• Proyek di Github
	• Dokumentasi SDK
	• Contoh Notebook berikut, yang tersedia di SageMaker konsol dan GitHub proyek terkait.
	  • `hello_world_workflow.ipynb`
	  • `machine_learning_workflow_abalone.ipynb`
	  • `training_pipeline_pytorch_mnist.ipynb`
		7 November 2019

Perubahan	Deskripsi	Tanggal diubah
Perbarui	Step Functions sekarang mendukung lebih banyak tindakan API untuk Amazon SageMaker, dan menyertakan dua proyek sampel baru untuk mendemonstrasikan fungsionalitas. Untuk informasi selengkapnya, lihat hal berikut. • Kelola SageMaker dengan Step Functions • Menggunakan AWS Step Functions dengan layanan lain • Melatih Model Machine Learning • Setel Model Machine Learning	3 Oktober 2019
Fitur baru	Step Functions mendukung memulai eksekusi alur kerja baru dengan memanggil `StartExecution` sebagai API layanan terintegrasi. Lihat: • Mulai Eksekusi Alur Kerja dari Status Tugas. • Kelola AWS Step Functions Eksekusi sebagai Layanan Terpadu • Menggunakan AWS Step Functions dengan layanan lain • Kebijakan IAM untuk Memulai Eksekusi Alur Kerja Step Functions	12 Agustus 2019

Perubahan	Deskripsi	Tanggal diubah
Fitur baru	Step Functions meliputi kemampuan untuk meneruskan token tugas ke layanan terintegrasi, dan menjeda eksekusi hingga token tugas dikembalikan dengan `SendTaskSuccess` atau `SendTaskFailure`. Lihat: • Pola integrasi layanan • Tunggu Panggilan Balik dengan Token Tugas • Contoh Pola Panggilan Balik (Amazon SQS, Amazon SNS, Lambda) • Integrasi yang dioptimalkan untuk Step Functions • Men-deploy Contoh Proyek Persetujuan Manusia • Metrik Integrasi Layanan Step Functions sekarang menyediakan cara untuk mengakses informasi dinamis tentang eksekusi Anda saat ini secara langsung di kolom `"Parameters"` ketetapan status. Lihat: • Objek konteks • Meneruskan Simpul Objek Konteks sebagai Parameter	23 Mei 2019
Fitur baru	Step Functions mendukung CloudWatch Events untuk perubahan status eksekusi, lihat: • EventBridge (CloudWatch Events) untuk perubahan status eksekusi Step Functions • Panduan Pengguna CloudWatch Acara Amazon	8 Mei 2019
Fitur baru	Step Functions mendukung izin IAM menggunakan tanda. Untuk informasi selengkapnya, lihat: • Penandaan di Step Functions • Kebijakan berbasis tanda	5 Maret 2019

Perubahan	Deskripsi	Tanggal diubah
Fitur baru	Step Functions Lokal sekarang tersedia. Anda dapat menjalankan Step Functions pada mesin lokal Anda untuk pengujian dan pengembangan. Step Functions Lokal tersedia untuk diunduh baik sebagai aplikasi Java, atau sebagai citra Docker. Lihat Menguji mesin negara secara lokal.	4 Februari 2019
Fitur baru	AWS Step Functions sekarang tersedia di wilayah Beijing dan Ningxia. Lihat Wilayah yang didukung.	15 Januari 2018
Fitur baru	Step Functions mendukung penandaan sumber daya untuk membantu melacak alokasi biaya Anda. Anda dapat menandai mesin status pada halaman Detail, atau melalui tindakan API. Lihat Penandaan di Step Functions.	7 Januari 2019
Fitur baru	AWS Step Functions sekarang tersedia di wilayah Eropa (Paris), dan Amerika Selatan (São Paulo). Lihat Wilayah yang didukung.	13 Desember 2018
Fitur baru	AWS Step Functions sekarang tersedia wilayah Eropa (Stockholm). Lihat Wilayah yang didukung untuk daftar wilayah yang didukung.	12 Desember 2018
Fitur baru	Step Functions sekarang terintegrasi dengan beberapa AWS layanan. Anda sekarang dapat langsung memanggil dan meneruskan parameter ke API layanan terintegrasi ini dari status tugas di Amazon States Language. Untuk informasi selengkapnya, lihat: • Menggunakan AWS Step Functions dengan layanan lain • Meneruskan parameter ke API layanan • Integrasi yang dioptimalkan untuk Step Functions	29 November 2018

Perubahan	Deskripsi	Tanggal diubah
Pembaruan	Peningkatan deskripsi `TimeoutSeconds` dan `HeartbeatSeconds` dalam dokumentasi untuk status tugas. Lihat Tugas.	24 Oktober 2018
Pembaruan	Peningkatan deskripsi untuk batasan Ukuran riwayat eksekusi maksimal dan menyediakan tautan ke topik praktik terbaik terkait. • Kuota yang berkaitan dengan eksekusi mesin status • Hindari mencapai kuota sejarah	17 Oktober 2018
Perbarui	Ditambahkan tutorial baru untuk AWS Step Functions dokumentasi: LihatMemulai Eksekusi Mesin Status dalam Respons terhadap Peristiwa Amazon S3.	25 September 2018
Pembaruan	Menghapus entri Eksekusi maksimum ditampilkan di konsol Step Functions dari dokumentasi batas. Lihat Kuota.	13 September 2018
Perbarui	Menambahkan topik praktik terbaik ke AWS Step Functions dokumentasi tentang peningkatan latensi saat melakukan polling untuk tugas aktivitas. Lihat Hindari latensi saat polling untuk tugas aktivitas.	30 Agustus 2018
Perbarui	Memperbaiki AWS Step Functions topik tentang kegiatan dan aktivitas pekerja. Lihat Aktivitas.	29 Agustus 2018
Perbarui	Memperbaiki AWS Step Functions topik tentang CloudTrail integrasi. Lihat Mencatat Step Functions Menggunakan AWS CloudTrail.	7 Agustus 2018
Perbarui	Menambahkan contoh JSON ke AWS CloudFormation tutorial. Lihat Membuat mesin status Lambda untuk Step Functions menggunakan AWS CloudFormation.	23 Juni 2018

Perubahan	Deskripsi	Tanggal diubah
Pembaruan	Penambahan topik baru tentang penanganan kesalahan layanan Lambda. Lihat Menangani pengecualian layanan Lambda.	20 Juni 2018
Fitur baru	AWS Step Functions sekarang tersedia di wilayah Asia Pasifik (Mumbai). Lihat Wilayah yang didukung untuk daftar wilayah yang didukung.	28 Juni 2018
Fitur baru	AWS Step Functions sekarang tersedia wilayah AWS GovCloud (AS-Barat). Lihat Wilayah yang didukung untuk daftar wilayah yang didukung. Untuk informasi tentang penggunaan Step Functions di Wilayah AWS GovCloud (AS-Barat), lihat AWS GovCloud (US).	28 Juni 2018
Pembaruan	Peningkatan dokumentasi penanganan kesalahan untuk status `Parallel`. Lihat Penanganan Kesalahan.	20 Juni 2018
Pembaruan	Peningkatan dokumentasi tentang pemrosesan Input dan Output di Step Functions. Pelajari cara menggunakan `InputPath` , `ResultPath` , dan `OutputPath` untuk mengontrol aliran JSON melalui alur kerja, status, dan tugas Anda. Lihat: • Pengolahan Input dan Output di Step Functions • ResultPath	7 Juni 2018
Pembaruan	Peningkatan contoh kode untuk status paralel. Lihat Paralel.	4 Juni 2018
Fitur baru	Anda sekarang dapat memantau metrik API dan Layanan di CloudWatch. Lihat Memantau Step Functions Menggunakan CloudWatch.	25 Mei 2018

Perubahan	Deskripsi	Tanggal diubah
Pembaruan	`StartExecution` , `StopExecution` , dan `StateTransition` sekarang telah meningkatkan batas throttling di wilayah berikut: • AS Timur (Virginia Utara) • US West (Oregon) • Eropa (Irlandia) Untuk informasi selengkapnya lihat Kuota.	16 Mei 2018
Fitur baru	AWS Step Functions sekarang tersedia wilayah AS Barat (California N.) dan Asia Pasifik (Seoul). Lihat Wilayah yang didukung untuk daftar wilayah yang didukung.	5 Mei 2018
Pembaruan	Pembaruan prosedur dan citra untuk mencocokkan perubahan pada antarmuka.	25 April 2018
Pembaruan	Penambahan tutorial baru yang menunjukkan cara memulai eksekusi baru untuk melanjutkan tugas Anda. Lihat Melanjutkan Eksekusi Alur Kerja yang Berjalan Lama sebagai Eksekusi Baru. Tutorial ini menjelaskan pola desain yang dapat membantu menghindari beberapa keterbatasan layanan. Lihat Hindari mencapai kuota sejarah.	19 April 2018
Pembaruan	Peningkatan pengenalan dokumentasi status dengan menambahkan informasi konseptual tentang mesin status. Lihat Status.	9 Maret 2018

Perubahan	Deskripsi	Tanggal diubah
Perbarui	Selain HTML, PDF, dan Kindle, Panduan AWS Step Functions Pengembang tersedia di GitHub. Untuk meninggalkan umpan balik, pilih GitHub ikon di sudut kanan atas. 	2 Maret 2018
Pembaruan	Penambahan topik yang menjelaskan sumber daya lain yang berkaitan dengan Step Functions. Lihat Informasi terkait.	20 Februari 2018

Fitur baru	• Ketika membuat mesin status baru, Anda harus memahami bahwa AWS Step Functions akan membuat IAM role yang memungkinkan akses ke fungsi Lambda Anda.
	• Pembaruan tutorial berikut untuk mencerminkan perubahan minor di alur kerja pembuatan mesin status:
	  • Membuat mesin status Step Functions yang menggunakan Lambda
	  • Membuat mesin status Aktivitas menggunakan Step Functions
	  • Menangani kondisi kesalahan menggunakan mesin status Step Functions
	  • Ulangi loop dengan Lambda
		19 Februari 2018

Perubahan	Deskripsi	Tanggal diubah
Pembaruan	Penambahan topik yang menjelaskan contoh pekerja aktivitas yang ditulis dalam Ruby. Implementasi ini dapat digunakan untuk membuat aktivitas pekerja Ruby secara langsung, atau sebagai pola desain untuk membuat pekerja aktivitas dalam bahasa lain. Lihat Contoh Activity Worker di Ruby.	6 Februari 2018
Pembaruan	Penambahan tutorial baru yang menjelaskan pola desain yang menggunakan fungsi Lambda untuk iterasi jumlah. Lihat Membuat mesin status Step Functions yang menggunakan Lambda.	31 Januari 2018
Pembaruan	Pembaruan konten pada izin IAM untuk menyertakan API `DescribeStateMachineForExecution` dan `UpdateStateMachine` . Lihat Membuat Izin IAM Terperinci untuk Pengguna Non-Admin.	26 Januari 2018
Pembaruan	Penambakan wilayah yang baru tersedia: Canada (Central) , Asia Pacific (Singapore). Lihat Wilayah yang didukung.	25 Januari 2018
Pembaruan	Pembaruan tutorial dan prosedur untuk mencerminkan bahwa IAM memungkinkan Anda untuk memilih Step Functions sebagai peran.	24 Januari 2018
Pembaruan	Penambahan topik Praktik Terbaik baru yang menunjukkan tidak meneruskan muatan besar antar status. Lihat Gunakan ARN Amazon S3 bukan meneruskan muatan besar.	23 Januari 2018

Perubahan	Deskripsi	Tanggal diubah
Pembaruan	Perbaikan prosedur untuk mencocokkan antarmuka diperbarui untuk membuat mesin status: • Membuat mesin status Step Functions yang menggunakan Lambda • Membuat mesin status Aktivitas menggunakan Step Functions • Menangani kondisi kesalahan menggunakan mesin status Step Functions	17 Januari 2018
Fitur Baru	Anda dapat menggunakan Proyek Sampel untuk menyediakan mesin status dan semua sumber daya AWS dengan cepat. LihatProyek sampel untuk Step Functions, Proyek sampel yang tersedia meliputi: • Polling untuk Status Pekerjaan (Lambda,) AWS Batch • Timer Tugas (Lambda, Amazon SNS) **Note** Proyek sampel ini dan dokumentasi terkait menggantikan tutorial yang menjelaskan implementasi fungsionalitas yang sama.	11 Januari 2018
Pembaruan	Penambahan bagian Praktik Terbaik yang mencakup informasi tentang menghindari eksekusi macet. Lihat Praktik terbaik untuk Step Functions.	5 Januari 2018

Perubahan	Deskripsi	Tanggal diubah
Pembaruan	Penambahan catatan tentang cara percobaan ulang dapat mempengaruhi harga: **Note** Percobaan ulang diperlakukan sebagai transisi status. Untuk informasi tentang cara transisi status mempengaruhi penagihan, lihat Harga Step Functions.	8 Desember 2017
Pembaruan	Penambahan informasi terkait dengan nama sumber daya: **Note** Step Functions memungkinkan Anda membuat nama untuk mesin status, eksekusi, dan aktivitas , serta label yang berisi karakter non-ASCII. Nama-nama non-ASCII ini tidak berfungsi dengan Amazon. CloudWatch Untuk memastikan bahwa Anda dapat melacak CloudWatch metrik, pilih nama yang hanya menggunakan karakter ASCII.	6 Desember 2017
Pembaruan	Peningkatan informasi gambaran umum keamanan dan penambahan topik tentang izin IAM terperinci. Lihat Keamanan di AWS Step Functions dan Membuat Izin IAM Terperinci untuk Pengguna Non-Admin.	27 November 2017
Fitur Baru	Anda dapat memperbarui mesin status yang ada. Lihat Memperbarui mesin status Anda.	15 November 2017

Perubahan	Deskripsi	Tanggal diubah
Pembaruan	Penambahan catatan untuk memperjelas kesalahan `Lambda.Unknown` dan terkait dengan dokumentasi Lambda di bagian berikut: • Nama kesalahan • Langkah 3: Buat mesin status dengan bidang Catch **Note** Kesalahan tidak tertangani di Lambda dilaporkan sebagai `Lambda.Unknown` dalam output kesalahan. Ini termasuk out-of-memory kesalahan dan batas waktu fungsi. Anda dapat mencocokkan di `Lambda.Unknown` , `States.ALL` , atau `States.TaskFailed` untuk menangani kesalahan ini. Ketika Lambda mencapai jumlah maksimum permintaan, kesalahannya adalah `Lambda.TooManyRequestsException` . Untuk informasi selengkapnya tentang kesalahan fungsi Lambda, lihat Penanganan kesalahan dan percobaan ulang otomatis di Panduan PengembangAWS Lambda .	17 Oktober 2017
Pembaruan	Koreksi dan klarifikasi instruksi IAM dan pembaruan tangkapan layar di semua tutorial.	11 Oktober 2017

Perubahan	Deskripsi	Tanggal diubah

Pembaruan	• Penambahan tangkapan layar baru untuk hasil eksekusi mesin status untuk mencerminkan perubahan di konsol Step Functions. Penulisan ulang instruksi Lambda dalam tutorial berikut untuk mencerminkan perubahan di konsol Lambda:
	  • Membuat mesin status Step Functions yang menggunakan Lambda
	  • Membuat Poller Status Tugas
	  • Membuat Timer Tugas
	  • Menangani kondisi kesalahan menggunakan mesin status Step Functions
	• Koreksi dan klarifikasi informasi tentang pembuatan mesin status di bagian berikut:
	  • Membuat mesin status Aktivitas menggunakan Step Functions
		6 Oktober 2017

Pembaruan	Penulisan ulang instruksi IAM di bagian berikut untuk mencerminkan perubahan di konsol IAM: • Membuat peran IAM untuk mesin negara Anda • Membuat mesin status Step Functions yang menggunakan Lambda • Membuat Poller Status Tugas • Membuat Timer Tugas • Menangani kondisi kesalahan menggunakan mesin status Step Functions • Membuat Step Functions API menggunakan API Gateway	5 Oktober 2017
Pembaruan	Penulisan ulang bagian Data Mesin Status.	28 September 2017

Perubahan	Deskripsi	Tanggal diubah
Fitur baru	Batas terkait dengan throttling tindakan API ditingkatkan untuk semua wilayah tempat Step Functions tersedia.	18 September 2017
Pembaruan	• Koreksi dan klarifikasi informasi tentang memulai eksekusi baru di semua tutorial. • Koreksi dan klarifikasi informasi di bagian Kuota yang terkait dengan akun.	14 September 2017
Pembaruan	Penulisan ulang tutorial berikut untuk mencerminkan perubahan di konsol Lambda: • Membuat mesin status Step Functions yang menggunakan Lambda • Menangani kondisi kesalahan menggunakan mesin status Step Functions • Membuat Poller Status Tugas	28 Agustus 2017
Fitur baru	Step Functions tersedia di Europe (London).	23 Agustus 2017
Fitur baru	Alur kerja visual mesin status memungkinkan Anda memperbesar, memperkecil, dan memusatkan grafik.	21 Agustus 2017

Perubahan	Deskripsi	Tanggal diubah
Fitur baru	**Important** Eksekusi tidak dapat menggunakan nama eksekusi lain selama 90 hari. Saat Anda melakukan beberapa `StartExecution` panggilan dengan nama yang sama, eksekusi baru tidak berjalan. Untuk informasi selengkapnya, lihat bagian parameter permintaan `name` tindakan API `StartExecution` di Referensi API AWS Step Functions .	18 Agustus 2017
Perbarui	Penambahan informasi tentang cara alternatif meneruskan ARN mesin status ke tutorial Membuat Step Functions API menggunakan API Gateway.	17 Agustus 2017
Pembaruan	Penambahan tutorial Membuat Poller Status Tugas baru.	10 Agustus 2017
Fitur baru	• Step Functions memancarkan `ExecutionThrottled` CloudWatch metrik. Untuk informasi selengkap nya, lihat Memantau Step Functions Menggunakan CloudWatch. • Penambahan bagian Kuota terkait throttling status.	3 Agustus 2017
Pembaruan	Pembaruan instruksi di bagian Langkah 1: Buat IAM role untuk API Gateway.	18 Juli 2017
Pembaruan	Koreksi dan klarifikasi informasi di bagian Pilihan.	23 Juni 2017

Perubahan	Deskripsi	Tanggal diubah
Perbarui	Menambahkan informasi tentang penggunaan sumber daya di bawah AWS akun lain ke tutorial berikut: • Membuat mesin status Step Functions yang menggunakan Lambda • Membuat mesin status Lambda untuk Step Functions menggunakan AWS CloudFormation • Membuat mesin status Aktivitas menggunakan Step Functions • Menangani kondisi kesalahan menggunakan mesin status Step Functions	22 Juni 2017
Pembaruan	Koreksi dan klarifikasi informasi di bagian berikut: • Menangani kondisi kesalahan menggunakan mesin status Step Functions • Status • Penanganan kesalahan dalam Step Functions	21 Juni 2017
Pembaruan	Penulisan ulang semua tutorial untuk mencocokkan penyegaran konsol Step Functions.	12 Juni 2017
Fitur baru	Step Functions tersedia di Asia Pacific (Sydney).	8 Juni 2017
Pembaruan	Restrukturisasi bagian Amazon States Language.	7 Juni 2017
Pembaruan	Koreksi dan klarifikasi informasi di bagian Membuat mesin status Aktivitas menggunakan Step Functions.	6 Juni 2017
Pembaruan	Koreksi contoh kode di bagian Nyatakan contoh mesin menggunakan Coba Ulang dan menggunakan Catch.	5 Juni 2017

Perubahan	Deskripsi	Tanggal diubah
Perbarui	Merestrukturisasi panduan ini menggunakan standar AWS dokumentasi.	31 Mei 2017
Pembaruan	Koreksi dan klarifikasi informasi di bagian Paralel.	25 Mei 2017
Pembaruan	Penggabungan bagian Jalur dan Filter ke bagian Pengolahan Input dan Output di Step Functions.	24 Mei 2017
Pembaruan	Koreksi dan klarifikasi informasi di bagian Memantau Step Functions Menggunakan CloudWatch.	15 Mei 2017
Pembaruan	Pembaruan kode pekerja `GreeterActivities.java` di tutorial Membuat mesin status Aktivitas menggunakan Step Functions.	9 Mei 2017
Pembaruan	Penambahan video pengantar bagian Apa itu AWS Step Functions?.	19 April 2017
Pembaruan	Koreksi dan klarifikasi informasi dalam tutorial berikut: • Membuat mesin status Step Functions yang menggunakan Lambda • Membuat mesin status Aktivitas menggunakan Step Functions • Menangani kondisi kesalahan menggunakan mesin status Step Functions	19 April 2017
Pembaruan	Penambahan informasi tentang templat Lambda ke tutorial Membuat mesin status Step Functions yang menggunakan Lambda dan Menangani kondisi kesalahan menggunakan mesin status Step Functions.	6 April 2017

Perubahan	Deskripsi	Tanggal diubah
Pembaruan	Perubahan batasan "Ukuran data hasil atau input maksimum" ke "Ukuran data hasil atau input maksimum untuk tugas, status, atau eksekusi" (32.768 karakter). Untuk informasi selengkapnya, lihat Kuota yang berkaitan dengan eksekusi tugas.	31 Maret 2017
Fitur baru	• Step Functions mendukung eksekusi state machine dengan menyetel Step Functions sebagai target Amazon CloudWatch Events.	21 Maret 2017
Fitur baru	• Step Functions memungkinkan penanganan kesalahan fungsi Lambda sebagai metode penanganan kesalahan yang dipilih. • Pembaruan tutorial Menangani kondisi kesalahan menggunakan mesin status Step Functions dan bagian Penanganan kesalahan dalam Step Functions.	16 Maret 2017
Fitur baru	Step Functions tersedia di Europe (Frankfurt).	7 Maret 2017
Pembaruan	Reorganisasi topik dalam daftar isi dan pembaruan tutorial berikut: • Membuat mesin status Step Functions yang menggunakan Lambda • Membuat mesin status Aktivitas menggunakan Step Functions • Menangani kondisi kesalahan menggunakan mesin status Step Functions	23 Februari 2017
Fitur baru	• Halaman Mesin Status konsol Step Functions mencakup tombol Salin ke Baru dan Hapus. • Pembaruan tangkapan layar untuk mencocokkan perubahan konsol.	23 Februari 2017

Perubahan	Deskripsi	Tanggal diubah
Fitur baru	• Step Functions mendukung pembuatan API menggunakan API Gateway. • Penambahan tutorial Membuat Step Functions API menggunakan API Gateway.	14 Februari 2017
Fitur baru	• Step Functions mendukung integrasi dengan AWS CloudFormation. • Penambahan tutorial Membuat mesin status Lambda untuk Step Functions menggunakan AWS CloudFormation.	10 Februari 2017
Pembaruan	Klarifikasi perilaku saat ini bidang `ResultPath` dan `OutputPath` terkait status `Parallel`.	6 Februari 2017
Pembaruan	• Klarifikasi pembatasan penamaan mesin status dalam tutorial. • Koreksi beberapa contoh kode.	5 Januari 2017
Pembaruan	Pembaruan contoh fungsi Lambda untuk menggunakan model pemrograman terbaru.	9 Desember 2016
Rilis awal	Rilis awal AWS Step Functions.	1 Desember 2016

AWSGlosarium

Untuk AWS terminologi terbaru, lihat AWSglosarium di Referensi. Glosarium AWS

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.