aws

Guida per gli sviluppatori

AWS Step Functions

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

AWS Step Functions: Guida per gli sviluppatori

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

I marchi e il trade dress di Amazon non possono essere utilizzati in relazione ad alcun prodotto o servizio che non sia di Amazon, in alcun modo che possa causare confusione tra i clienti, né in alcun modo che possa denigrare o screditare Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà delle rispettive aziende, che possono o meno essere associate, collegate o sponsorizzate da Amazon.

Table of Contents

Che cos'è AWS Step Functions? 1
AWS SDK e integrazioni ottimizzate 2
Flussi di lavoro Standard ed Express 2
Specifiche dei flussi di lavoro standard 2
Specifiche dei flussi di lavoro Express 2
Casi d'uso 3
Caso d'uso #1: orchestrazione delle funzioni 3
Caso d'uso #2: Ramificazione 4
Caso d'uso #3: gestione degli errori 4
Caso d'uso #4: Human in the loop 5
Caso d'uso #5: elaborazione parallela 6
Caso d'uso #6: parallelismo dinamico 6
Integrazioni dei servizi 7
Regioni supportate 10
È la prima volta che usi Step Functions? 11
Prerequisiti 12
Fase 1: Creare un account e un utente IAM 12
Registrarsi per creare un Account AWS 12
Creazione di un utente amministratore 13
Fase 2: Concessione dell'accesso programmatico 14
Nozioni di base 16
Concetti chiave 16
Tutorial di questa serie 18
Tutorial 1: Crea il prototipo per la tua macchina a stati 22
Fasi successive 24
Tutorial 2: Definizione della prima integrazione del servizio utilizzando una funzione Lambda 24
Fase 1: Creare e testare la funzione Lambda 24
Passaggio 2: aggiorna il flusso di lavoro: configura lo stato Get credit limit 25
Passaggi successivi 26
Tutorial 3: Implementa una condizione if-else nel tuo flusso di lavoro 26
Fase 1: Crea un argomento Amazon SNS che riceva il token di callback 27
Passaggio 2: creare una funzione Lambda per gestire il callback 28
Passaggio 3: aggiorna il flusso di lavoro: aggiungi la logica delle condizioni if-else nello stato Choice 30

Fasi successive ... 33
Tutorial 4: Definisci più attività da eseguire in parallelo ... 33
Fase 1: Creare le funzioni Lambda per eseguire i controlli richiesti ... 33
Fase 2: Aggiornamento del flusso di lavoro: aggiungere attività parallele da eseguire ... 35
Tutorial 5: Iterazione simultanea su una raccolta di elementi ... 36
Fase 1: Creare una tabella DynamoDB per memorizzare il nome di tutte le agenzie di credito ... 37
Fase 2: Aggiornare la macchina a stati — Recupera i risultati dalla tabella DynamoDB ... 38
Fase 3: Creare una funzione Lambda che restituisca i punteggi di credito per tutte le agenzie di credito ... 39
Passaggio 4: Aggiorna la macchina a stati: aggiungi uno stato della mappa per recuperare in modo iterativo i punteggi di credito ... 39
Tutorial 6: Salvare il flusso di lavoro ed eseguire la macchina a stati ... 40
Fase 1: Salvare la macchina a stati ... 40
Passaggio 2: aggiungi le politiche IAM rimanenti ... 42
Fase 3: Esegui la macchina a stati ... 42
Tutorial 7: Configurazione di input e output ... 43
Seleziona porzioni specifiche dell'input grezzo utilizzando il InputPath filtro ... 45
Manipola l'input selezionato utilizzando il filtro Parametri ... 49
Configurare l'output utilizzando OutputPath i filtri ResultSelectorResultPath, e ... 49
Tutorial 8: errori di debug nella console ... 52
Effettuare il debug del percorso non valido Errore di stato della scelta ... 53
Eseguire il debug degli errori di espressione del percorso JSON durante l'applicazione di filtri di input e output ... 55
Casi d'uso ... 58
Elaborazione dei dati ... 58
Machine learning ... 60
Orchestrazione di microservizi ... 61
Automazione IT e della sicurezza ... 62
Come funziona Step Functions ... 64
Flussi di lavoro Standard e Express ... 64
Flussi di lavoro Express sincroni e asincroni ... 68
Garanzie di esecuzione ... 69
Ottimizzazione dei costi con Express Workflows ... 71
States ... 73
Amazon States Language ... 75

Pass ... 97
Attività ... 98
Choice ... 120
Attendi ... 126
Succeed ... 128
Fail ... 129
Parallel ... 131
Eseguire la mappatura ... 136
Modalità di elaborazione dello stato della mappa ... 136
Differenze tra modalità in linea e modalità distribuita ... 137
Utilizzo dello stato della mappa in modalità Inline ... 139
Utilizzo dello stato della mappa in modalità distribuita ... 148
Soglia di errore tollerata per lo stato della mappa distribuita ... 159
Transizioni ... 162
Transizioni nello stato della mappa distribuita ... 163
Dati delle macchine a stati ... 163
Formato dei dati ... 164
Input/output delle macchine a stati ... 164
Input/output degli stati ... 165
Elaborazione di input e output ... 166
Percorsi ... 169
InputPath, Parametri e ResultSelector ... 171
ResultPath ... 177
OutputPath ... 186
InputPathResultPath, ed OutputPath esempi ... 187
Campi di input e output dello stato della mappa ... 192
Oggetto Context ... 225
Simulatore di flusso di dati ... 231
Utilizzo del simulatore di flusso di dati ... 232
Considerazioni sul simulatore di flusso di dati ... 233
Versioni e alias ... 235
Versioni ... 236
Alias ... 240
Autorizzazione per versioni e alias ... 243
Associazione di esecuzioni a una versione o a un alias ... 245
Esempio di implementazione ... 249

Distribuzione graduale delle versioni 252
Esecuzioni 261
Avvio delle esecuzioni da un'attività 262
Usare Scheduler EventBridge 264
Esecuzioni Standard ed Express Workflow 270
Visualizzazione ed esecuzione del debug delle esecuzioni 276
Redrivingesecuzioni 298
Esame di Map Run 308
Gestione degli errori 321
Nomi degli errori 322
Riprovare dopo un errore 325
Stati di fallback 329
Esempi di macchine a stati che utilizzano Retry e Catch 331
Invoca Step Functions 336
Consistenza di lettura 337
Inserimento di tag in Step Functions 337
Tagging per l'allocazione dei costi 338
Tagging per la sicurezza 339
La visualizzazione e gestione di 340
API Tagging 340
Workflow Studio 341
Panoramica dell'interfaccia 342
Modalità di progettazione 342
Modalità codice 349
modalità Config 353
Tasti di scelta rapida 357
Utilizzo di Workflow Studio 358
Crea un flusso di lavoro 358
Progetta un flusso di lavoro 360
Esegui il tuo flusso di lavoro 367
Modifica il tuo flusso di lavoro 368
Esporta il flusso di lavoro 370
Crea il prototipo del tuo flusso di lavoro 371
Configura input e output 372
Configurare l'input in uno stato 373
Configura l'output di uno stato 376

Ruoli di esecuzione in Workflow Studio 381
Informazioni sui ruoli generati automaticamente 382
Generazione automatica di ruoli 382
Risoluzione dei problemi relativi alla generazione di ruoli 384
Ruolo per il test delle attività HTTP in Workflow Studio 385
Ruolo per testare un'integrazione di servizi ottimizzata in Workflow Studio 385
Ruolo per testare l'integrazione di un servizio AWS SDK in Workflow Studio 386
Ruolo per testare gli stati del flusso in Workflow Studio 386
Gestione degli errori 387
Riprova sugli errori 388
Rileva gli errori 389
Timeout 389
HeartbeatSeconds 389
Tutorial: Impara a usare AWS Step Functions Workflow Studio 390
Passaggio 1: accedi a Workflow Studio 391
Fase 2: Creare una macchina a stati 391
Fase 3: Rivedi la definizione di Amazon States Language generata automaticamente 393
Fase 4: Modificare la definizione del flusso di lavoro in modalità Codice 395
Fase 5: Salvare la macchina a stati 397
Passaggio 6: Eseguire la macchina a stati 398
Passaggio 7: aggiorna la tua macchina a stati 399
Fase 8: Pulizia 401
Tutorial 402
Creare una macchina a stati Step Functions che utilizza Lambda 402
Fase 1: Creazione di una funzione Lambda 403
Fase 2: testare la funzione Lambda 404
Passaggio 3: Creare una macchina a stati 405
Fase 4: Esegui la macchina a stati 407
Gestione delle condizioni di errore con una macchina a stati 409
Fase 1: Creare una funzione Lambda che fallisce 410
Fase 2: testare la funzione Lambda 411
Fase 3: Creare una macchina a stati con un campo Catch 411
Fase 4: Esegui la macchina a stati 414
Ripeti un'azione utilizzando lo stato della mappa in linea 415
Fase 1: Creare il prototipo del flusso di lavoro 416
Fase 2: Configurare input e output 416

Fase 3: rivedi la definizione di Amazon States Language generata automaticamente e salva il flusso di lavoro 418
Fase 4: Esegui la macchina a stati 420
Guida introduttiva all'utilizzo dello stato di Distributed Map 421
Prerequisiti 422
Passaggio 1: crea il prototipo del flusso di lavoro 422
Passaggio 2: configura i campi richiesti per lo stato della mappa 423
Passaggio 3: Configurare le opzioni aggiuntive 424
Fase 4: Configurare la funzione Lambda 425
Passaggio 5: aggiorna il prototipo del flusso di lavoro 426
Passaggio 6: rivedi la definizione di Amazon States Language generata automaticamente e salva il flusso di lavoro 426
Passaggio 7: Esegui la macchina a stati 429
Elaborazione di un intero batch di dati con una funzione Lambda 430
Fase 1: Creare la macchina a stati 430
Fase 2: Creare la funzione Lambda 432
Passaggio 3: Esegui la macchina a stati 433
Elaborazione di singoli elementi di dati con una funzione Lambda 435
Fase 1: Creare la macchina a stati 436
Fase 2: Creare la funzione Lambda 438
Passaggio 3: Esegui la macchina a stati 433
Avvio dell'esecuzione di una macchina a stati in risposta agli eventi di Amazon S3 443
Prerequisito: creazione di una macchina a stati 444
Fase 1: creare un bucket in Amazon S3 444
Fase 2: abilitare la notifica degli eventi di Amazon S3 con EventBridge 444
Passaggio 3: crea una EventBridge regola Amazon 445
Fase 4: test della regola 446
Esempio di input di esecuzione 447
Creazione di un'API Step Functions utilizzando API Gateway 448
Fase 1: creazione di un ruolo IAM per API Gateway 448
Fase 2: Crea la tua API API Gateway 449
Fase 3: Test e implementazione dell'API API Gateway 452
Creazione di una macchina a stati Step FunctionsAWS SAM 455
Prerequisiti 455
Passaggio 1: scaricare un'applicazione AWS SAM di esempio 456
Passaggio 2: crea la tua applicazione 457

Fase 3: Caricamento dell'applicazione inAWSCloud 458
Risoluzione dei problemi 459
Eliminare 460
Creazione di una macchina a stati Activity 461
Fase 1: creazione di un'attività 461
Passaggio 2: crea una macchina a stati 462
Fase 3: implementazione di un lavoratore 464
Fase 4: Eseguire la macchina a stati 466
Fase 5: esecuzione e arresto del lavoratore 467
Iterare un ciclo con Lambda 468
Passaggio 1: creare una funzione Lambda per iterare un conteggio 468
Fase 2: testare la funzione Lambda 469
Fase 3: creazione di una macchina a stati 471
Fase 4: avvio di una nuova esecuzione 473
Continuazione del lavoro in corso come nuova esecuzione 474
Utilizzo di un'azione API Step Functions (consigliato) 475
Utilizzo di una funzione Lambda 479
Distribuire un progetto di approvazione umana di esempio 491
Fase 1: crea un modello 492
Fase 2: Creare una pila 492
Fase 3: approvare l'abbonamento SNS 493
Passaggio 4: Esegui la macchina a stati 494
Codice di origine del modello 496
Visualizza le tracce X-Ray in Step Functions 506
Fase 1: creare un ruolo IAM per Lambda 507
Passaggio 2: creazione di una funzione Lambda 507
Fase 3: Creare altre due funzioni Lambda 509
Fase 4: Creare una macchina a stati 509
Passaggio 5: Esegui la macchina a stati 512
Raccogli informazioni sui bucket Amazon S3 utilizzando AWS le integrazioni dei servizi SDK .. 515
Fase 1: Creare la macchina a stati 515
Passaggio 2: aggiungi le autorizzazioni necessarie per il ruolo IAM 518
Passaggio 3: Esegui l'esecuzione di una macchina a stati standard 518
Fase 4: Eseguire l'esecuzione di una macchina a stati Express 519
Strumenti per sviluppatori 521
Opzioni di sviluppo 521

Console Step Functions ... 522
AWS SDK ... 522
Flussi di lavoro Standard ed Express ... 523
API del servizio HTTPS ... 523
Ambienti di sviluppo ... 523
Endpoints ... 524
AWS CLI ... 524
Step Functions Local ... 524
AWS Toolkit for Visual Studio Code ... 525
AWS Serverless Application Model e Step Functions ... 525
Terraform e Step Functions ... 525
Supporto per il formato di definizione ... 526
Step Functions e AWS SAM ... 533
Perché usare Step Functions con AWS SAM? ... 534
Integrazione di Step Functions con le AWS SAM specifiche ... 534
Integrazione delle funzioni a fasi con l'interfaccia a riga di comando SAM ... 534
DefinitionSubstitutions nei modelli AWS SAM ... 535
Passaggi successivi ... 539
Utilizzo di Workflow Studio in Application Composer ... 540
Utilizzo di Workflow Studio in Application Composer ... 541
Fai riferimento dinamicamente alle risorse utilizzando sostituzioni di definizione CloudFormation ... 541
Connect le attività di integrazione dei servizi a schede componenti avanzate ... 541
Importa progetti esistenti e sincronizzali localmente ... 542
Funzionalità di Workflow Studio non disponibili in Strumento AWS per la creazione di applicazioni ... 543
Creazione di una macchina a stati Lambda utilizzando AWS CloudFormation ... 544
Passaggio 1: configura il AWS CloudFormation modello ... 544
Fase 2: Usa il AWS CloudFormation modello per creare una Lambda State Machine ... 549
Fase 3: Avviare l'esecuzione di una State Machine ... 555
Creazione di una macchina a Lambda stati utilizzando AWS CDK ... 555
Fase 1: Configurazione del progetto AWS CDK ... 556
Passaggio 2: Utilizzare AWS CDK per creare una macchina a stati ... 558
Fase 3: Avviare l'esecuzione di una macchina a stati ... 567
Fase 4: eliminazione ... 568
Passaggi successivi ... 568

Creazione di un'API Gateway REST con Synchronous Express State Machine utilizzando AWS CDK 569
Fase 1: Configura il tuo progetto AWS CDK 569
Passaggio 2: utilizzare AWS CDK per creare un'API API Gateway REST con integrazione backend Synchronous Express State Machine 573
Fase 3: Testare l'API Gateway 583
Fase 4: eliminazione 586
Data Science SDK 586
Distribuzione di macchine a stati utilizzando Terraform 587
Prerequisiti 587
Ciclo di vita di sviluppo con Terraform 588
Ruoli e politiche IAM per la tua macchina a stati 590
Esecuzione di test e debugging 592
Utilizzo dell' TestState API 592
Considerazioni sull'utilizzo dell'API TestState 593
Utilizzo dei livelli di ispezione nell'API TestState 594
IAMautorizzazioni per l'utilizzo dell'API TestState 601
Test di uno stato (console) 602
Testare uno stato utilizzando AWS CLI 603
Test e debug del flusso di dati di input e output 609
Test di macchine a stato locale 613
Configurazione di Step Functions Local (versione scaricabile) e Docker 614
Configurazione locale di Step Functions (versione scaricabile) - Versione Java 615
Impostazione delle opzioni di configurazione per Step Functions Local 616
Esecuzione di Step Functions Local sul computer 618
Test di Step Functions e AWS SAM CLI Local 620
Utilizzo di integrazioni di servizi simulati 625
Best practice 643
Utilizza i timeout per evitare esecuzioni bloccate 643
Usa gli ARN Amazon S3 invece di trasferire payload di grandi dimensioni 644
Evita di raggiungere la quota di cronologia 646
Gestire le eccezioni del servizio Lambda 647
Evita la latenza durante i sondaggi relativi alle attività 648
Scelta di flussi di lavoro Standard o Express 649
Limitazioni relative alle dimensioni della politica delle risorse di Amazon CloudWatch Logs 650
Uso di altri servizi 651

Chiama altri servizi AWS 651
Integrazioni ottimizzate 652
AWS Integrazioni SDK 652
Supporto per modelli di integrazione 652
Accesso multi-account 655
AWS Integrazioni di servizi SDK 656
Utilizzo delle integrazioni dei servizi SDK AWS 656
Servizi supportati 658
Azioni API non supportate per i servizi supportati 700
Integrazioni di servizi SDK obsolete AWS 702
Integrazioni ottimizzate 702
Amazon API Gateway 706
Amazon Athena 714
AWS Batch 716
Amazon Bedrock 718
AWS CodeBuild 722
Amazon DynamoDB 728
Amazon ECS/Fargate 731
Amazon EKS 734
Amazon EMR 749
Amazon EMR su EKS 761
Amazon EMR Serverless 765
Amazon EventBridge 774
AWS Glue 776
AWS Glue DataBrew 777
AWS Lambda 778
AWS Elemental MediaConvert 782
Amazon SageMaker 785
Amazon SNS 796
Amazon SQS 799
AWS Step Functions 801
Chiama API di terze parti 805
Definizione del task HTTP 806
Campi HTTP Task 806
Autenticazione per un task HTTP 813
Unione dei dati di EventBridge connessione e di definizione delle attività HTTP 814

Applicazione della codifica URL sul corpo della richiesta ... 817
Autorizzazioni IAM per eseguire un task HTTP ... 819
Esempio di attività HTTP ... 820
Test di un task HTTP ... 822
Risposte HTTP Task non supportate ... 824
Modelli di integrazione dei servizi ... 825
Richiesta e risposta ... 825
Esegui un processo (.sync) ... 826
Attendere un callback con il token dell'attività ... 828
Passa i parametri a un'API di servizio ... 834
Passa JSON statico come parametri ... 834
Passa l'input di stato come parametri usando Paths ... 835
Passa nodi di oggetti contestuali come parametri ... 836
Registro delle modifiche per le integrazioni ... 836
Progetti di esempio per Step Functions ... 860
Gestire un processo batch (AWS Batch,Amazon SNS) ... 861
Fase 1: Creare la macchina a stati e fornire risorse ... 861
Fase 2: Eseguire la macchina a stati ... 864
Esempio di codice della macchina a stati ... 865
Esempio IAM ... 866
Gestire un'attività relativa al contenitore (Amazon ECS,Amazon SNS) ... 867
Fase 1: Creare la macchina a stati e fornire risorse ... 867
Fase 2: Eseguire la macchina a stati ... 869
Esempio di codice della macchina a stati ... 870
Esempio IAM ... 872
Trasferisci record di dati (Lambda,DynamoDB,Amazon SQS) ... 873
Fase 1: Creare la macchina a stati e fornire risorse ... 874
Fase 2: Eseguire la macchina a stati ... 876
Esempio di codice della macchina a stati ... 877
Esempio IAM ... 879
Sondaggio per Job Status (AWS Batch Lambda,) ... 880
Fase 1: Creare la macchina a stati e fornire risorse ... 881
Fase 2: Eseguire la macchina a stati ... 883
Esempio di codice della macchina a stati ... 886
Timer delle attività (Lambda, Amazon SNS) ... 887
Fase 1: Creare la macchina a stati e fornire risorse ... 888

Fase 2: Eseguire la macchina a stati 890
Esempio di pattern di callback (Amazon SQS, Amazon SNS, Lambda) 892
Fase 1: Creare la macchina a stati e fornire risorse 893
Fase 2: Eseguire la macchina a stati 895
Esempio di callback Lambda 897
Gestisci un lavoro in Amazon EMR 897
Fase 1: creazione della State Machine e fornitura di risorse 898
Fase 2: Eseguire la macchina a stati 869
Esempio di codice della macchina a stati 870
Esempio IAM 872
Esegui un EMR Serverless lavoro 906
Modello AWS CloudFormation e risorse aggiuntive 907
Fase 1: Creare la macchina a stati e fornire risorse 907
Fase 2: Eseguire la macchina a stati 909
Avvio di un workflow all'interno di un workflow (Step Functions, Lambda) 910
Fase 1: Creare la macchina a stati e fornire risorse 911
Fase 2: Eseguire la macchina a stati 913
Esempio di codice della macchina a stati 914
Elabora dinamicamente i dati con uno stato della mappa 916
Fase 1: Creare la macchina a stati e fornire risorse 917
Fase 2: Abbonarsi all'argomento Amazon SNS 920
Fase 3: Aggiungere messaggi alla coda Amazon SQS 920
Fase 4: Esegui la macchina a stati 921
Esempio di codice di macchina a stati 922
Esempio IAM 924
Elabora un file CSV con Distributed Map 925
AWS CloudFormation modello e risorse aggiuntive 926
Fase 1: Creare la macchina a stati e fornire risorse 927
Fase 2: Eseguire la macchina a stati 930
Elabora i dati in un bucket Amazon S3 con Distributed Map 931
AWS CloudFormation modello e risorse aggiuntive 932
Fase 1: Creare la macchina a stati e fornire risorse 933
Fase 2: Eseguire la macchina a stati 936
Training su un modello di Machine Learning 937
Fase 1: Creazione della macchina a stati e fornitura delle risorse 937
Fase 2: Eseguire la macchina a stati 940

Esempio di codice della macchina a stati ... 941
Esempio IAM ... 943
Ottimizzazione di un modello di Machine Learning. ... 944
Fase 1: Creazione della macchina a stati e fornitura delle risorse ... 945
Fase 2: Eseguire la macchina a stati ... 947
Esempio di codice della macchina a stati ... 949
Esempi IAM ... 953
Elabora messaggi ad alto volume da Amazon SQS (Express Workflows) ... 956
Fase 1: Creare la macchina a stati e fornire risorse ... 957
Fase 2: Attivare l'esecuzione della macchina a stati ... 959
Esempio di codice di funzione Lambda ... 960
Esempio di codice della macchina a stati ... 961
Esempio IAM ... 962
Esempio di checkpoint selettivo (flussi di lavoro Express) ... 963
Fase 1: Creare la State Machine e fornire risorse ... 964
Fase 2: Eseguire la macchina a stati ... 966
Codice macchina a stati di esempio per il padre (flussi di lavoro standard) ... 968
Esempio di ruolo IAM per la Parent State Machine ... 970
Codice macchina a stati di esempio per la macchina a stati nidificato (flussi di lavoro Express) ... 968
Esempio di ruolo IAM per Child State Machine ... 974
Crea un AWS CodeBuild progetto (CodeBuild, Amazon SNS) ... 975
Fase 1: Creare la macchina a stati e fornire risorse ... 975
Fase 2: Eseguire la macchina a stati ... 978
Esempio di codice della macchina a stati ... 979
Preelabora i dati e addestra un modello di machine learning ... 980
Fase 1: Creazione della macchina a stati e fornitura delle risorse ... 981
Fase 2: Eseguire la macchina a stati ... 983
Esempio di codice della macchina a stati ... 985
Esempio IAM ... 988
Esempio di orchestrazione Lambda ... 989
Fase 1: Creare la macchina a stati e fornire risorse ... 990
Fase 2: Eseguire la macchina a stati ... 992
Informazioni sulla macchina a stati e sulla sua esecuzione ... 994
Esempi IAM ... 997
Avvia una query su Athena ... 999

Fase 1: Creazione della macchina a stati e fornitura delle risorse 1000
Fase 2: Eseguire la macchina a stati 1002
Esempio di codice della macchina a stati 1003
Esempio IAM 1005
Esegui più query (Amazon Athena, Amazon SNS) 1007
Fase 1: Creare la macchina a stati e fornire risorse 1007
Fase 2: Eseguire la macchina a stati 1011
Esempio di codice della macchina a stati 1011
Esempi IAM 1014
Interroga set di dati di grandi dimensioni (Amazon Athena, Amazon S3 e Amazon SNS AWS Glue) 1018
Fase 1: Creare la macchina a stati e fornire risorse 1018
Fase 2: Eseguire la macchina a stati 1021
Esempio di codice della macchina a stati 1022
Esempi IAM 1024
Mantieni aggiornati i dati (Amazon Athena, Amazon S3,) AWS Glue 1027
Fase 1: Creare la macchina a stati e fornire risorse 1027
Fase 2: Eseguire la macchina a stati 1029
Esempio di codice della macchina a stati 1030
Esempio IAM 1032
Gestisci un cluster Amazon EKS 1034
Fase 1: Creare la macchina a stati e fornire risorse 1034
Fase 2: Eseguire la macchina a stati 1037
Esempio di codice della macchina a stati 1038
Esempio IAM 1042
Effettuare una chiamata a API Gateway 1044
Fase 1: Creare la macchina a stati e fornire risorse 1044
Fase 2: Eseguire la macchina a stati 1046
Esempio di codice della macchina a stati 1047
Esempio IAM 1049
Chiama un microservizio con API Gateway 1050
Fase 1: Creazione della macchina a stati e fornitura delle risorse 1050
Fase 2: Eseguire la macchina a stati 1052
Esempio di codice della macchina a stati 1054
Esempio IAM 1055
Invia un evento personalizzato a EventBridge 1057

Fase 1: Creazione della macchina a stati e fornitura delle risorse 1057
Fase 2: Eseguire la macchina a stati 1059
Esempio di codice della macchina a stati 1060
Esempio IAM 1061
Richiama flussi di lavoro Synchronous Express 1062
Fase 1: Creazione della macchina a stati e fornitura delle risorse 1062
Fase 2: Eseguire la macchina a stati 1065
Esempio di codice della macchina a stati 1066
Esempi IAM 1068
Esegui flussi di lavoro ETL/ELT con Amazon Redshift 1069
Fase 1: Creazione della macchina a stati e fornitura delle risorse 1070
Fase 2: Eseguire la macchina a stati 1072
Esempio di codice della macchina a stati 1074
Esempio IAM 1094
Utilizzo Step Functions e AWS Batch con gestione degli errori 1095
Fase 1: Creare la macchina a stati e fornire risorse 1095
Fase 2: Eseguire la macchina a stati 1097
Esempio di codice della macchina a stati 1098
Esempio IAM 1100
Trova un AWS Batch lavoro 1101
Fase 1: Creare la macchina a stati e fornire risorse 1101
Fase 2: Eseguire la macchina a stati 1104
Esempio di codice della macchina a stati 1105
Esempio IAM 1106
AWS Batch con Lambda 1107
Fase 1: Creazione della macchina statale e fornitura di risorse 1107
Fase 2: Eseguire la macchina a stati 1109
Esempio di codice della macchina a stati 1111
Esempio IAM 1112
Esegui il concatenamento dei prompt AI con Amazon Bedrock 1113
Modello AWS CloudFormation e risorse aggiuntive 1114
Prerequisiti 1114
Fase 1: Creare la macchina a stati e fornire risorse 1114
Fase 2: Eseguire la macchina a stati 1117
Quote 1119
Quote generali 1120

Quote relative agli account 1121
Quote relative a HTTP Task 1122
Quote relative alla limitazione statale 1122
Quote relative alla limitazione delle azioni delle API 1123
Quota relativa all' TestState API 1124
Altre quote 1125
Quote relative alle esecuzioni di macchine a stati 1128
Quote relative all'esecuzione delle attività 1131
Quote relative a versioni e alias 1132
Restrizioni relative all'etichettatura 1132
Registrazione e monitoraggio 1134
CloudWatch Metriche Amazon 1134
Metriche che segnalano un intervallo di tempo 1135
Metriche che riportano un conteggio 1136
Parametri di esecuzione 1136
Metriche del conteggio delle risorse per versioni e alias 1139
Parametri delle attività 1140
Parametri della funzione Lambda 1141
Parametri di integrazione del servizio 1142
Parametri servizio 1143
Parametri API 1144
Fornitura delle metriche Best-Effort CloudWatch 1145
Visualizzazione delle metriche per Step Functions 1145
Impostazione degli allarmi per Step Functions 1147
EventBridge Eventi Amazon 1150
EventBridge carichi utili 1151
Esempi di eventi Step Functions 1151
Instradamento di un evento Step Functions a EventBridge 1156
Funzioni della fase di registrazione tramite AWS CloudTrail 1157
Informazioni su Step Functions in CloudTrail 1158
Esempio: voci del file di registro di Step Functions 1158
Registrazione utilizzandoCloudWatchRegistri 1164
Configurazione della registrazione 1164
CloudWatchRegistra i payload 1165
Policy IAM per l'accesso aCloudWatchRegistri 1165
Livelli di log 1167

X-Ray 1170
Installazione e configurazione 1172
Concetti 1176
Integrazioni dei servizi 1177
Visualizzazione della console X-Ray 1178
Visualizzazione delle informazioni di tracciamento X-Ray per Step Functions 1179
Tracce 1179
Mappa servizio 1179
Segmenti e sottosegmenti 1181
Analisi 1183
Configurazione 1184
Cosa succede se non ci sono dati nella mappa di traccia o nella mappa dei servizi? 1185
Utilizzo Notifiche all'utente AWS con Step Functions 1185
Sicurezza 1186
Protezione dei dati 1186
Crittografia 1187
Identity and Access Management 1187
Destinatari 1188
Autenticazione con identità 1188
Gestione dell'accesso con policy 1192
Controllo degli accessi 1195
Operazioni di policy 1195
Risorse di policy 1196
Chiavi di condizione delle policy 1197
ACL 1197
ABAC 1198
Credenziali temporanee 1198
Autorizzazioni del principale 1199
Ruoli di servizio 1199
Ruoli collegati ai servizi 1200
Come AWS Step Functions funziona con IAM 1200
Esempi di policy basate su identità 1201
Policy basate su identità 1204
Policy basate su risorse 1205
AWS politiche gestite 1205
Creazione di un ruolo IAM per una macchina a stati 1207

Creazione di autorizzazioni IAM granulari per utenti non amministratori 1210
Accesso alle risorse di più account AWS 1213
Endpoint VPC 1224
Politiche IAM per servizi integrati 1227
Politiche IAM per l'utilizzo dello stato della mappa distribuita 1318
Policy basate su tag 1324
Risoluzione dei problemi 1325
Registrazione e monitoraggio 1327
Convalida della conformità 1327
Resilienza 1328
Sicurezza dell'infrastruttura 1329
Analisi della configurazione e delle vulnerabilità 1329
Migrazione dei carichi di lavoro da AWS Data Pipeline 1330
Migrazione dei carichi di lavoro 1330
Mappatura concettuale 1331
Progetti di esempio di Step Functions 1332
Confronto dei prezzi 1333
Risoluzione dei problemi 1334
Risoluzione dei problemi generali 1334
Non riesco a creare una macchina a stati. 1334
Non riesco a usare a JsonPath per fare riferimento all'output dell'attività precedente. 1334
C'è stato un ritardo nelle transizioni di stato. 1335
Quando avvio nuove esecuzioni di Standard Workflow, queste falliscono con l'`ExecutionLimitExceeded`errore. 1335
Un errore su un ramo in uno stato parallelo causa il fallimento dell'intera esecuzione. 1335
Risoluzione dei problemi di integrazione dei servizi 1336
Il mio lavoro è completo nel servizio a valle, ma in Step Functions lo stato dell'attività rimane «In corso» o il suo completamento è ritardato. 1336
Voglio restituire un output JSON dall'esecuzione di una macchina a stati annidati. 1336
Non riesco a richiamare una funzione Lambda da un altro account. 1336
Non riesco a vedere i task token trasmessi dagli `.waitForTaskToken` stati. 1338
Attività di risoluzione dei problemi 1338
L'esecuzione della mia macchina a stati è bloccata in uno stato di attività. 1338
Il mio assistente in attività scade in attesa di un task token. 1339
Risoluzione dei problemi dei flussi di lavoro Express 1339

La mia applicazione si interrompe prima di ricevere una risposta da una chiamata StartSyncExecution API. 1339
Non riesco a visualizzare la cronologia delle esecuzioni per risolvere gli errori di Express Workflow. 1339
Informazioni correlate 1341
Lancio recente di funzionalità 1342
Cronologia dei documenti 1345
Glossario per AWS 1390
..... mcccxci

Che cos'è AWS Step Functions?

AWS Step Functions è un servizio di orchestrazione senza server che consente l'integrazione con AWS Lambdafunzioni e altro Servizi AWS per creare applicazioni aziendali critiche. Attraverso la console grafica di Step Functions, puoi vedere il flusso di lavoro dell'applicazione come una serie di passaggi guidati dagli eventi.

Step Functions si fonda su attività e macchine a stati. In Step Functions, un flusso di lavoro viene chiamato macchina a stati, che consiste in una serie di passaggi basati sugli eventi. Ogni fase di un flusso di lavoro è denominata stato. Uno Attività stato rappresenta un'unità di lavoro svolta da un altro AWS servizio, ad esempio. AWS Lambda Uno stato Task può chiamare Servizio AWS qualsiasi API.

Con i controlli integrati di Step Functions, esamini lo stato di ogni fase del flusso di lavoro per assicurarti che l'applicazione funzioni nell'ordine e come previsto. A seconda del caso d'uso, è possibile disporre di AWS servizi di chiamata Step Functions, come Lambda, per eseguire attività. È possibile creare flussi di lavoro che elaborano e pubblicano modelli di machine learning. È possibile disporre di AWS servizi di controllo Step Functions AWS Glue, ad esempio per creare flussi di lavoro di estrazione, trasformazione e caricamento (ETL). È possibile anche creare flussi di lavoro automatizzati e di lunga durata per applicazioni che richiedono l'interazione umana.

> **Tip**
>
> Per familiarizzare con le funzionalità principali di Step Functions attraverso una serie di moduli interattivi, consulta The AWS Step Functions Workshop. Oppure inizia a utilizzare Step Functions seguendo questi tutorial introduttivi per creare un flusso di lavoro per l'applicazione di carte di credito.

Argomenti

- AWS SDK e integrazioni ottimizzate
- Flussi di lavoro Standard ed Express
- Casi d'uso
- Integrazioni dei servizi
- Regioni supportate
- È la prima volta che usi Step Functions?

AWS SDK e integrazioni ottimizzate

Per chiamare altri AWS servizi, puoi utilizzare le integrazioni AWS SDK di Step Functions oppure puoi utilizzare una delle integrazioni ottimizzate di Step Functions.

- Le integrazioni AWS SDK ti consentono di chiamare uno qualsiasi degli oltre duecento AWS servizi direttamente dalla tua macchina a stati, dandoti accesso a oltre novemila azioni API.
- Le integrazioni ottimizzate di Step Functions sono state personalizzate per semplificare l'utilizzo nelle vostre macchine a stati.

Flussi di lavoro Standard ed Express

Step Functions ha due tipi di flussi di lavoro. I flussi di lavoro standard prevedono un'esecuzione esatta del flusso di lavoro e possono durare fino a un anno. Ciò significa che ogni fase di un flusso di lavoro Standard verrà eseguita esattamente una volta. I flussi di lavoro Express, tuttavia, prevedono l'esecuzione del at-least-once flusso di lavoro e possono durare fino a cinque minuti. Ciò significa che uno o più passaggi in un Express Workflow possono potenzialmente essere eseguiti più di una volta, mentre ogni passaggio del flusso di lavoro viene eseguito almeno una volta.

Le esecuzioni sono casi in cui si esegue il flusso di lavoro per eseguire attività. I flussi di lavoro standard sono ideali per flussi di lavoro di lunga durata e verificabili, in quanto mostrano la cronologia di esecuzione e il debug visivo. I flussi di lavoro Express sono ideali per high-event-rate carichi di lavoro, come l'elaborazione di dati in streaming e l'ingestione di dati IoT.

Specifiche dei flussi di lavoro standard

- Frequenza di esecuzione di 2.000 al secondo
- Tasso di transizione tra stati di 4.000 al secondo
- Prezzo per transizione statale
- Mostra la cronologia di esecuzione e il debug visivo
- Supporta tutte le integrazioni e i modelli di servizio

Specifiche dei flussi di lavoro Express

- Frequenza di esecuzione di 100.000 al secondo

- Tasso di transizione statale quasi illimitato
- Prezzo in base al numero e alla durata delle esecuzioni
- Invia la cronologia di esecuzione ad Amazon CloudWatch
- Mostra la cronologia di esecuzione e il debug visivo in base al livello di log abilitato
- Supporta tutte le integrazioni di servizi e la maggior parte dei modelli

Per ulteriori informazioni sui flussi di lavoro Standard ed Express, inclusi i prezzi di Step Functions, consulta quanto segue:

- Flussi di lavoro Standard e Express
- Prezzi di AWS Step Functions

Casi d'uso

Step Functions gestisce i componenti e la logica dell'applicazione, così puoi scrivere meno codice e concentrarti sulla creazione e sull'aggiornamento rapido dell'applicazione. Questa sezione descrive i casi d'uso tipici per lavorare con Step Functions.

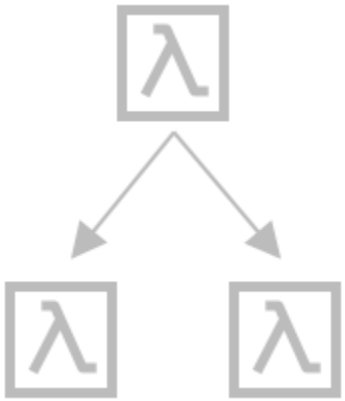
Caso d'uso #1: orchestrazione delle funzioni

Si crea un flusso di lavoro che esegue un gruppo di funzioni Lambda (passaggi) in un ordine specifico. L'output di una funzione Lambda passa all'input della funzione Lambda successiva. L'ultima fase del flusso di lavoro produce un risultato. Con Step Functions, puoi vedere come ogni fase del tuo flusso di lavoro interagisce tra loro, in modo da assicurarti che ogni passaggio svolga la funzione prevista.

Per un tutorial che mostra come creare una macchina a stati con un gruppo di funzioni, consulta quanto segue:

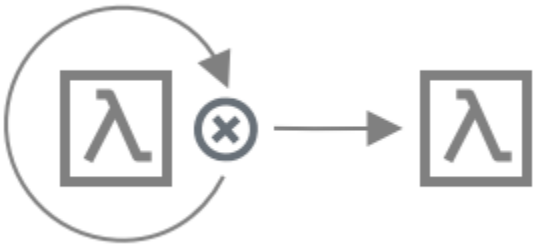
- Nozioni di base su AWS Step Functions

Caso d'uso #2: Ramificazione



Un cliente richiede un aumento del limite di credito. Utilizzando uno `Choice` stato, puoi fare in modo che Step Functions prenda decisioni in base all'input `Choice` dello stato. Se la richiesta supera il limite di credito preapprovato dal cliente, puoi fare in modo che Step Functions invii la richiesta del cliente a un responsabile per l'approvazione. Se la richiesta è inferiore al limite di credito preapprovato dal cliente, puoi fare in modo che Step Functions approvi la richiesta automaticamente.

Caso d'uso #3: gestione degli errori



`Retry`

In questo caso d'uso, un cliente richiede un nome utente. La prima volta, la richiesta del cliente non va a buon fine. Utilizzando una `Retry` dichiarazione, puoi fare in modo che Step Functions riprovi la richiesta del cliente. La seconda volta, la richiesta del cliente ha esito positivo.

`Catch`

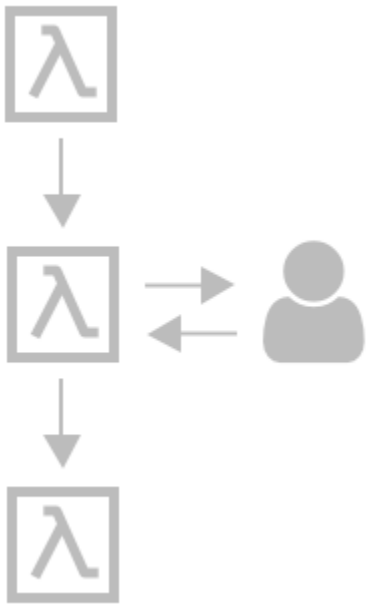
In un caso d'uso simile, un cliente richiede un nome utente non disponibile. Utilizzando un'`Catch`istruzione, Step Functions suggerisce un nome utente disponibile. Se il cliente utilizza il nome utente disponibile, puoi fare in modo che Step Functions passi alla fase successiva del flusso di lavoro, ovvero l'invio di un'e-mail di conferma. Se il cliente non utilizza il nome utente disponibile, devi

fare in modo che Step Functions passi a una fase diversa del tuo flusso di lavoro, ovvero ricominciare la procedura di registrazione.

Per esempi `Retry` e `Catch` dichiarazioni più dettagliati, consulta quanto segue:

- Gestione degli errori in Step Functions

Caso d'uso #4: Human in the loop

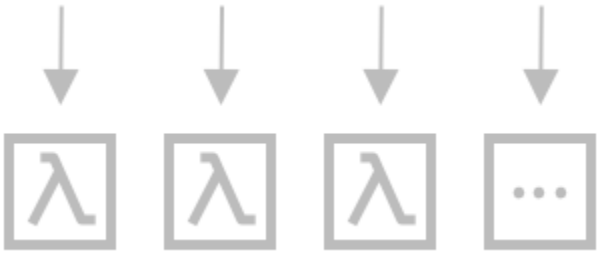


Utilizzando un'app bancaria, uno dei tuoi clienti invia denaro a un amico. Il cliente attende un'e-mail di conferma. Con un callback e un task token, puoi fare in modo che Step Functions dica a Lambda di inviare il denaro del cliente e di riferire quando un amico del cliente lo riceve. Dopo che Lambda avrà comunicato che un amico del cliente ha ricevuto il denaro, puoi fare in modo che Step Functions passi alla fase successiva del tuo flusso di lavoro, ovvero inviare al cliente un'email di conferma.

Per vedere un progetto di esempio che mostra un callback con un token task, consulta quanto segue:

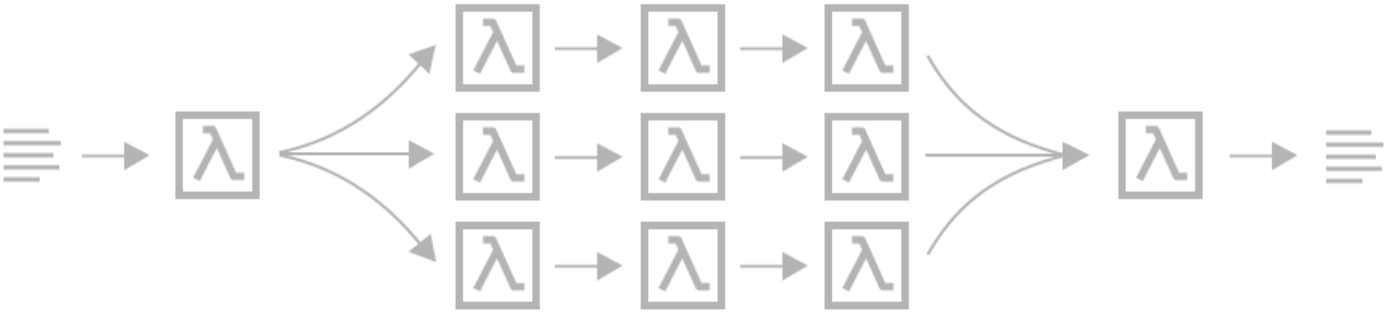
- Esempio di pattern di callback (Amazon SQS, Amazon SNS, Lambda)

Caso d'uso #5: elaborazione parallela



Un cliente converte un file video in cinque diverse risoluzioni di visualizzazione, in modo che gli spettatori possano guardarlo su più dispositivi. Utilizzando uno `Parallel` stato, Step Functions inserisce il file video, in modo che Lambda possa elaborarlo contemporaneamente nelle cinque risoluzioni dello schermo.

Caso d'uso #6: parallelismo dinamico



Un cliente ordina tre articoli e tu devi preparare ogni articolo per la consegna. Verifica la disponibilità di ogni articolo, ritira ogni articolo e poi impacchetta ogni articolo per la consegna. Utilizzando uno `Map` stato, Step Functions consente a Lambda di elaborare in parallelo ogni articolo del cliente. Una volta che tutti gli articoli del cliente sono stati imballati per la consegna, Step Functions passa alla fase successiva del flusso di lavoro, che consiste nell'inviare al cliente un'e-mail di conferma con le informazioni di tracciamento.

Per vedere un progetto di esempio che mostra il parallelismo dinamico utilizzando uno `Map` stato, consulta quanto segue:

- Elabora dinamicamente i dati con uno stato della mappa

Integrazioni dei servizi

Step Functions si integra con più AWS servizi. Per combinare Step Functions con questi servizi, utilizzate i seguenti modelli di integrazione dei servizi:

Richiedi una risposta (impostazione predefinita)

- Chiama un servizio e lascia che Step Functions passi allo stato successivo dopo aver ricevuto una risposta HTTP.

Esegui un lavoro (.sync)

- Chiama un servizio e chiedi a Step Functions di attendere il completamento di un lavoro.

Attendi una richiamata con un task token (. waitForTaskToken)

- Chiama un servizio con un task token e fai in modo che Step Functions attenda che il task token ritorni con una callback.

La tabella seguente mostra le integrazioni di servizio disponibili e i modelli di integrazione dei servizi per Step Functions.

I flussi di lavoro standard e i flussi di lavoro Express supportano le stesse integrazioni ma non gli stessi modelli di integrazione.

- Il supporto dei modelli di integrazione ottimizzati è diverso per ogni integrazione.
- I flussi di lavoro Express non supportano Run a Job (.sync) o Wait for Callback (. waitForTaskToken).
- Per ulteriori informazioni, consulta Flussi di lavoro Standard e Express.

Standard Workflows

Integrazioni di servizi supportate

	Servizio	Richiesta e risposta	Esegui un job (`.sync`)	Attendi il callback (`.waitForTaskToken`)
Integrazioni ottimizzate	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	✓	✓	
	Amazon SageMaker	✓	✓	

	Servizio	Richiesta e risposta	Esegui un job (`.sync`)	Attendi il callback (`.waitForTaskToken`)
	Amazon SNS	✓		✓
	Amazon SQS	✓		✓
	AWS Step Functions	✓	✓	✓
AWS Integrazioni SDK	Oltre duecento	✓		✓

Express Workflows

Integrazioni di servizi supportate

	Servizio	Richiesta e risposta	Esegui un job (`.sync`)	Attendi il callback (`.waitForTaskToken`)
Integrazioni ottimizzate	Amazon API Gateway	✓		
	Amazon Athena	✓		
	AWS Batch	✓		
	Amazon Bedrock	✓		
	AWS CodeBuild	✓		
	Amazon DynamoDB	✓		
	Amazon ECS/Fargate	✓		
	Amazon EKS	✓		

	Servizio	Richiesta e risposta	Esegui un job (`.sync`)	Attendi il callback (`.waitForTaskToken`)
	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 Integrazioni SDK	Oltre duecento	✓		

Regioni supportate

La maggior parte AWS delle regioni supporta Step Functions. Per un elenco completo delle AWS regioni in cui è disponibile Step Functions, consulta la Tabella delle AWS regioni.

È la prima volta che usi Step Functions?

Se è la prima volta che utilizzi Step Functions, i seguenti argomenti ti aiutano a comprendere le diverse parti dell'utilizzo di Step Functions, incluso il modo in cui Step Functions si combina con altri AWS servizi:

- Tutorial per Step Functions
- Progetti di esempio per Step Functions
- AWS Step Functions SDK per Data Science per Python

Prerequisiti per iniziare con AWS Step Functions

Prima di iniziare AWS Step Functions per la prima volta, completa i prerequisiti elencati in questa pagina.

Argomenti

- Fase 1: Registrati per diventare un Account AWS utente IAM
- Fase 2: Concessione dell'accesso programmatico

Fase 1: Registrati per diventare un Account AWS utente IAM

Per accedere a qualsiasi servizio AWS, è necessario innanzitutto creare un Account AWS.
Puoi utilizzare il tuo Account AWS per visualizzare le attività e i report di utilizzo e per gestire l'autenticazione e l'accesso. Ti vengono addebitati solo i prodotti e i servizi che utilizzi e puoi iniziare a utilizzare AWS gratuitamente. Per ulteriori informazioni, consulta Piano gratuito AWS.

Per evitare di utilizzare le azioni Utente root dell'account AWS for Step Functions, è consigliabile creare un utente IAM per ogni persona che necessita dell'accesso amministrativo a Step Functions.

Se hai già un AWS account, passa al prerequisito successivo.

Registrarsi per creare un Account AWS

Se non disponi di un Account AWS, completa la procedura seguente per crearne uno.

Per registrarsi a un Account AWS

1. Apri la pagina all'indirizzo https://portal.aws.amazon.com/billing/signup.
2. Segui le istruzioni online.

   Nel corso della procedura di registrazione riceverai una telefonata, durante la quale sarà necessario inserire un codice di verifica attraverso la tastiera del telefono.

   Durante la registrazione di un Account AWS, viene creato un Utente root dell'account AWS. L'utente root dispone dell'accesso a tutte le risorse e tutti i Servizi AWSnell'account. Come best practice di sicurezza, assegna l'accesso amministrativo a un utente amministrativo e utilizza solo l'utente root per eseguire attività che richiedono l'accesso di un utente root.

Al termine del processo di registrazione, riceverai un'e-mail di conferma da AWS. È possibile visualizzare l'attività corrente dell'account e gestire l'account in qualsiasi momento accedendo all'indirizzo https://aws.amazon.com/ e selezionando Il mio account.

Creazione di un utente amministratore

Dopo esserti registratoAccount AWS, proteggi Utente root dell'account AWSAWS IAM Identity Center, abilita e crea un utente amministrativo in modo da non utilizzare l'utente root per le attività quotidiane.

Protezione dell'Utente root dell'account AWS

1. Accedi alla AWS Management Console come proprietario dell'account scegliendo Utente root e immettendo l'indirizzo email del Account AWS. Nella pagina successiva, inserisci la password.

   Per informazioni sull'accesso utilizzando un utente root, consulta la pagina Accesso come utente root della Guida per l'utente di Accedi ad AWS.
2. Abilita l'autenticazione a più fattori (MFA) per l'utente root.

   Per ricevere istruzioni, consulta Abilitazione di un dispositivo MFA virtuale per l'utente root dell'Account AWS (console) nella Guida per l'utente IAM.

Creazione di un utente amministratore

1. Abilita IAM Identity Center.

   Per istruzioni, consulta Enabling AWS IAM Identity Center nella Guida AWS IAM Identity Center per l'utente.
2. In IAM Identity Center, concedi l'accesso amministrativo a un utente amministrativo.

   Per un tutorial sull'utilizzo di IAM Identity Center directory come fonte di identità, consulta Configurare l'accesso utente con le impostazioni predefinite IAM Identity Center directory nella Guida per l'AWS IAM Identity Centerutente.

Accesso come utente amministratore

- Per accedere con l'utente IAM Identity Center, utilizza l'URL di accesso che è stato inviato al tuo indirizzo e-mail quando hai creato l'utente IAM Identity Center.

Per informazioni sull'accesso utilizzando un utente IAM Identity Center, consulta Accedere al portale di accesso AWS nella Guida per l'utente Accedi ad AWS.

Fase 2: Concessione dell'accesso programmatico

Gli utenti hanno bisogno di un accesso programmatico se desiderano interagire con AWS esternamente a AWS Management Console. La modalità con cui concedere l'accesso programmatico dipende dal tipo di utente che accede ad AWS.

Per fornire agli utenti l'accesso programmatico, scegli una delle seguenti opzioni.

Quale utente necessita dell'accesso programmatico?	Per	Come
Identità della forza lavoro (Utenti gestiti nel centro identità IAM)	Utilizza credenziali temporane e per firmare richieste programmatiche alla AWS CLI, agli SDK AWS o alle API AWS.	Segui le istruzioni per l'interfa ccia che desideri utilizzare. • Per la AWS CLI, consulta la pagina Configurazione della AWS CLI per l'uso di AWS IAM Identity Center nella Guida per l'utente dell'AWS Command Line Interface. • Per gli SDK AWS, gli strumenti e le API AWS, consulta la pagina Autentica zione Centro identità IAM nella Guida di riferimento per SDK e strumenti AWS.
IAM	Utilizza credenziali temporane e per firmare richieste programmatiche alla AWS CLI, agli SDK AWS o alle API AWS.	Segui le istruzioni in Utilizzo di credenziali temporanee con le risorse AWS nella Guida per l'utente IAM.

Quale utente necessita dell'accesso programmatico?	Per	Come
IAM	(Non consigliato) Utilizza credenziali a lungo termine per firmare richieste programmatiche alla AWS CLI, agli SDK AWS o alle API AWS.	Segui le istruzioni per l'interfaccia che desideri utilizzare. • Per la AWS CLI, consulta la pagina Autenticazione tramite credenziali utente IAM nella Guida per l'utente dell'AWS Command Line Interface. • Per gli SDK e gli strumenti AWS, consulta la pagina Autenticazione con credenziali a lungo termine nella Guida di riferimento per SDK e strumenti AWS. • Per le API AWS, consulta la pagina Gestione delle chiavi di accesso per utenti IAM nella Guida per l'utente IAM.

Nozioni di base su AWS Step Functions

Benvenuto in Step FunctionsGuida introduttivaserie di tutorial.

Step Functions è un servizio di orchestrazione serverless che consente di la definizione di un flusso di lavoro applicativo come una serie di passaggi serverless. Ogni fase del flusso di lavoro è denominatastato. Si utilizzano più comunemente stati, ad esempioAttività,Choice,Parallel, eEseguire la mappatura, per definire i flussi di lavoro. All'internoTaskstati, puoi usare ilAWSIntegrazioni SDK supportate da Step Functions e che orchestrano piùServizi AWSnei tuoi flussi di lavoro.

Argomenti

- Concetti chiave
- Tutorial di questa serie
- Tutorial 1: Crea il prototipo per la tua macchina a stati
- Tutorial 2: Definizione della prima integrazione del servizio utilizzando una funzione Lambda
- Tutorial 3: Implementa una condizione if-else nel tuo flusso di lavoro
- Tutorial 4: Definisci più attività da eseguire in parallelo
- Tutorial 5: Iterazione simultanea su una raccolta di elementi
- Tutorial 6: Salvare il flusso di lavoro ed eseguire la macchina a stati
- Tutorial 7: Configurazione di input e output
- Tutorial 8: errori di debug nella console

Concetti chiave

Questa sezione presenta i concetti più importanti di Step Functions. Prima di iniziare, esamina i seguenti concetti chiave.

Termine	Descrizione
Flusso di lavoro	Descrive una sequenza di passaggi e spesso corrisponde a un processo aziendale.

Termine	Descrizione
Workflow Studio	Un designer visivo di flussi di lavoro che ti aiuta a prototipare e creare flussi di lavoro più velocemente. Per ulteriori informazioni, consulta AWS Step Functions Workflow Studio.
States	Singoli passaggi nella macchina a stati, che eseguono una varietà di funzioni nella macchina a stati. Per ulteriori informazioni, consulta States.
Macchine a stati	Un flusso di lavoro definito utilizzando testo JSON che rappresenta i singoli stati o passaggi del flusso di lavoro insieme a campi, come`StartAt`,`TimeoutSeconds` , `eVersion`. Per ulteriori informazioni, consulta Struttura della macchina a stati.
Amazon States Language	Un linguaggio strutturato basato su JSON utilizzato per la definizione della macchina a stati. È una raccolta distatiche può funzionare (`Task`stato), determina a quali stati passare a quello successivo (`Choice`stato) e interrompe un'esecuzione con un errore (`Fail`stato). Per ulteriori informazioni, consulta Amazon States Language.
Configurazione di input e output	I singoli stati di un flusso di lavoro ricevono i dati JSON come input e in genere passano i dati JSON come output allo stato successivo. Step Functions fornisce più filtri per controllare il flusso di dati in ingresso e in uscita tra gli stati. Per ulteriori informazioni, consulta Elaborazione di input e output in Step Functions.
Integrazione dei servizi	Step Functions si integra direttamente conServizi AWS, consentendoti di richiamare le azioni API di ciascun servizio dal tuo flusso di lavoro. Per ulteriori informazioni, consulta Utilizzo AWS Step Functions con altri servizi.

Termine	Descrizione
Tipo di integrazione del servizio	Step Functions offre i seguenti tipi di integrazione dei servizi: • Integrazioni ottimizzate— Personalizzato da Step Functions per fornire funzionalità speciali per un flusso di lavoro. Ad esempioLambda Invokeconvertirà l'output dell'API da una stringa JSON in escape a un oggetto JSON. • AWSintegrazioni SDK— Si comporta esattamente come una chiamata API standard utilizzandoAWSSDK. Puoi chiamare uno degli oltre duecentoServizi AWSdirettamente dalla tua macchina a stati e accedi a oltre novemila azioni API. Per ulteriori informazioni, consulta Utilizzo AWS Step Functions con altri servizi.
Modello di integrazione dei servizi	Chiamare un integratoServizio AWSnel flusso di lavoro, si utilizza uno dei seguenti modelli di integrazione dei servizi forniti da Step Functions: • Richiedi una risposta (impostazione predefinita)— Chiama un servizio e lascia che Step Functions passi allo stato successivo dopo aver ricevuto una risposta HTTP. • Esegui un lavoro (.sync)— Chiama un servizio e chiedi a Step Functions di attendere il completamento di un lavoro. • Attendi una richiamata con un task token (.waitForTaskGettone)— Chiama un servizio con un task token e fai in modo che Step Functions attenda che il task token ritorni con una callback.
Esecuzione	Le esecuzioni di macchine a stati sono casi in cui si esegue il flusso di lavoro per eseguire attività. Per ulteriori informazioni, consulta Esecuzioni in Step Functions.

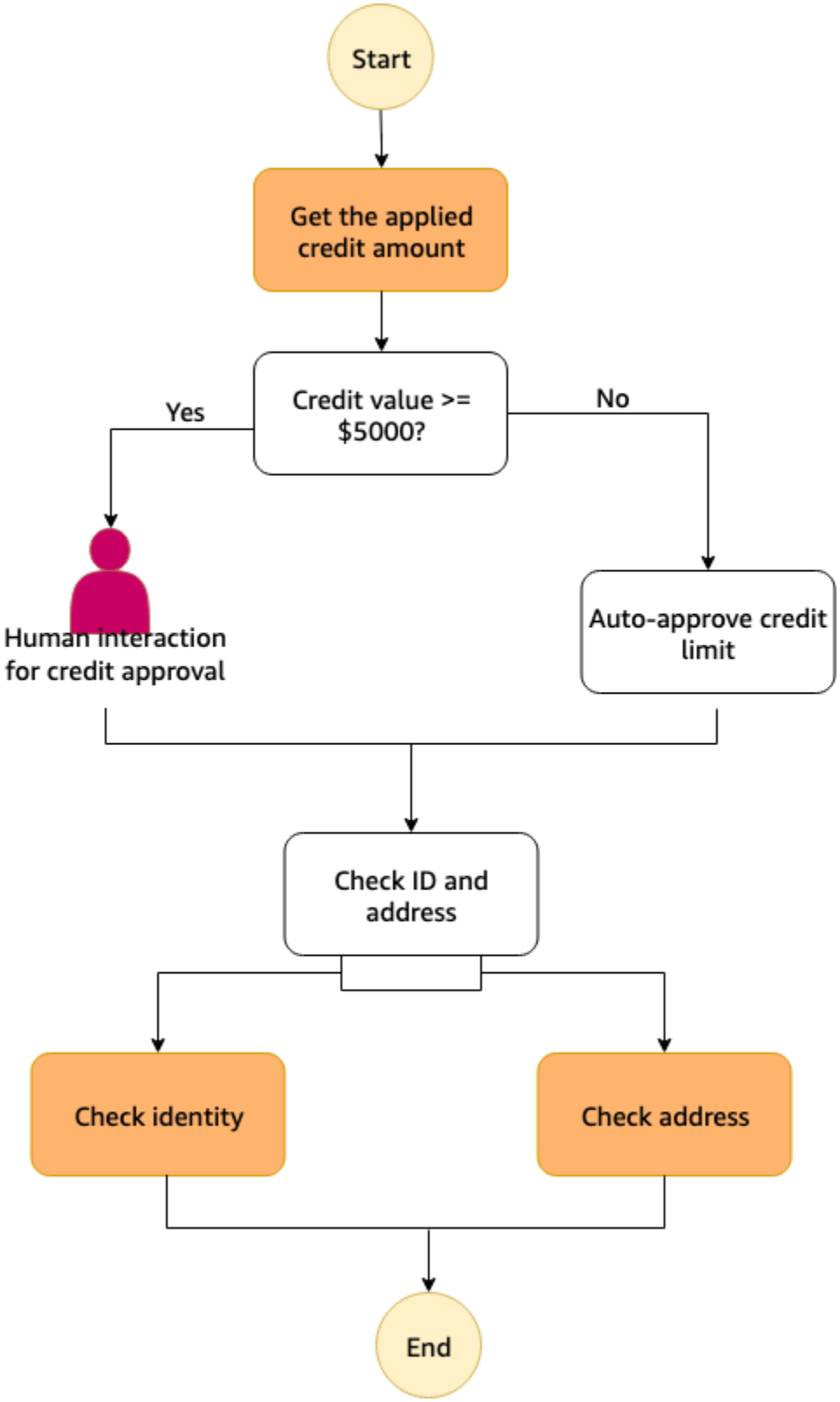
Tutorial di questa serie

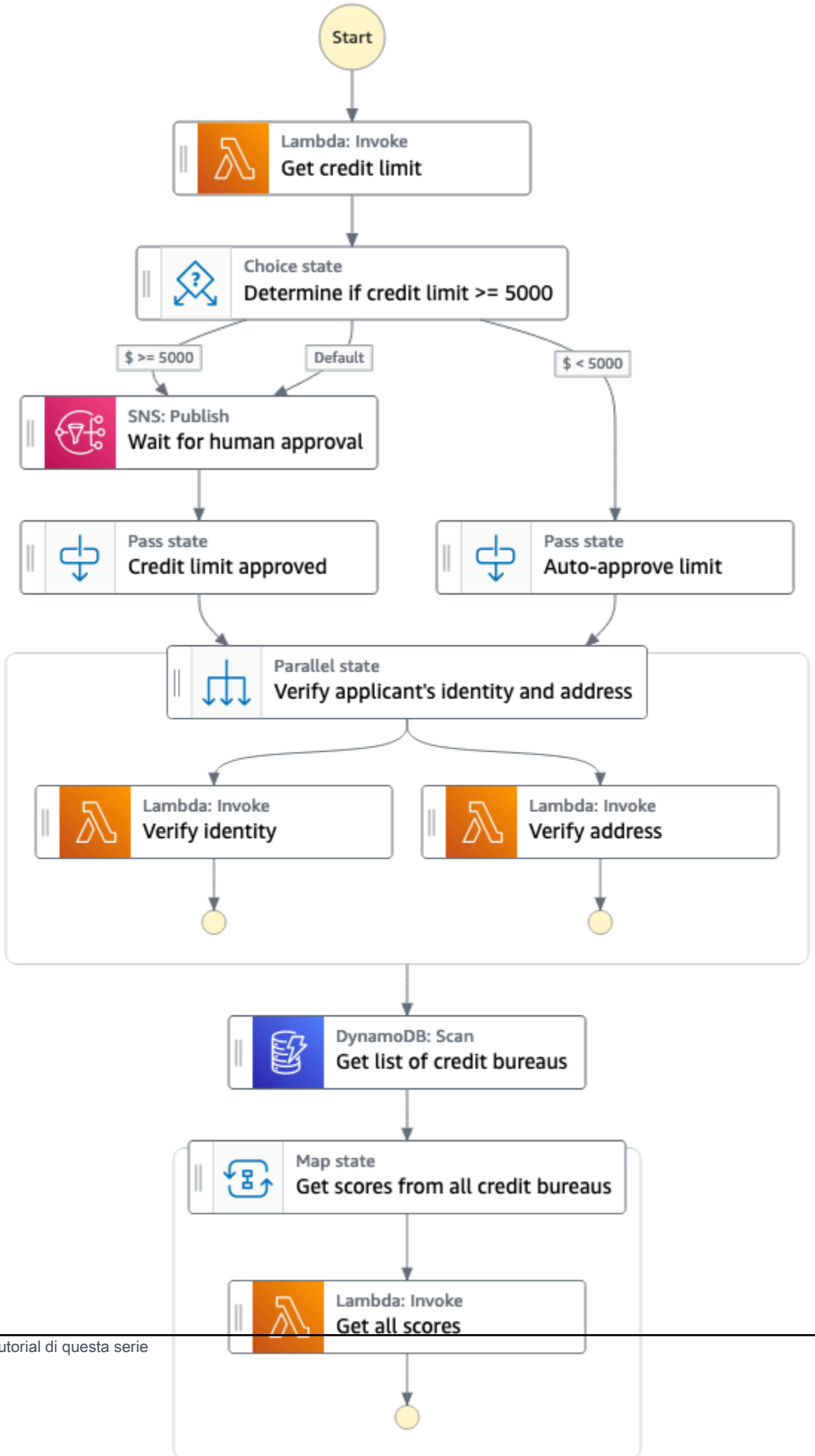
LaGuida introduttivai tutorial in questo capitolo illustrano la creazione di un flusso di lavoro di base per l'elaborazione delle richieste di carte di credito. In questi tutorial, imparerai a usare gli stati di uso comune in Step Functions. Integrerai il tuo flusso di lavoro con altriServizi AWS, ad esempioAWS

Lambdae Amazon Simple Notification Service. Dopo aver completato questi tutorial, avrai un semplice flusso di lavoro che simula l'elaborazione di una richiesta di carta di credito.

> **Note**
>
> Mentre questiGuida introduttivai tutorial descrivono il flusso di lavoro di un'applicazione di carte di credito, puoi usare Step Functions per creare più tipi di flussi di lavoro. Ad esempio, è possibile creare flussi di lavoro per l'elaborazione dei dati, l'automazione IT, l'apprendimento automatico, l'elaborazione dei media o l'elaborazione degli ordini.

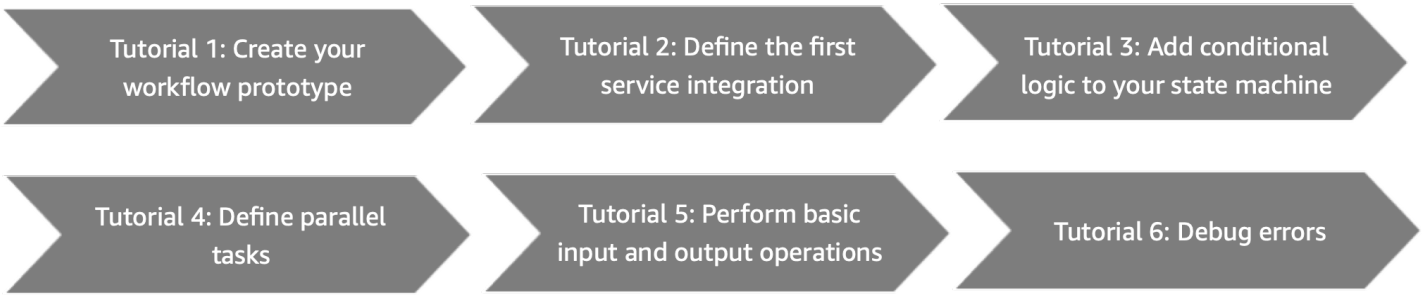
Le immagini seguenti rappresentano il flusso di lavoro di un'applicazione di carte di credito e il modo in cui appare quando viene orchestrato utilizzando Step Functions. Ogni fase del diagramma di flusso è rappresentata da uno stato nel flusso di lavoro Step Functions.


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**
>
> Ti consigliamo di completare tutti questi tutorial in ordine. Il completamento dei tutorial completi ti insegnerà come utilizzare concetti e funzionalità fondamentali per l'utilizzo di Step Functions nei flussi di lavoro di produzione.

La seguente tabella di marcia mostra i passaggi da eseguire per creare il flusso di lavoro di elaborazione delle carte di credito utilizzando Workflow Studio di Step Functions. Questi passaggi sono presentati come una serie di tutorial che contengono istruzioni su come completare tale passaggio.



Prima di iniziare, assicurati di completare ilprerequisiti.

Tutorial 1: Crea il prototipo per la tua macchina a stati

In questo tutorial, creerai il prototipo per il flusso di lavoro di elaborazione delle carte di credito utilizzandoWorkflow Studio di Step Functions. Sceglierai le azioni e gli stati dell'API richiesti dalAzionieFlussoseleziona rispettivamente le schede e utilizza la funzione drag and drop di Workflow Studio per creare il prototipo del flusso di lavoro. Nei tutorial successivi, imparerai come configurare ilServizi AWSe gli stati di Step Functions che utilizzerai in questo flusso di lavoro.

Per creare il prototipo della macchina a stati

1. ApriConsole Step Functionse scegliCrea macchina a stati.
2. NelScegli un modellofinestra di dialogo, selezionaVuota.
3. Selezionare Select (Seleziona). Questo apre Workflow Studio inModalità di progettazione.
4. In Workflow Studio, dalAzionischeda, trascinaAWS LambdaInvocaAzione API e rilasciala nello stato vuoto etichettatoTrascina qui il primo stato. Puoi lasciare indicata la configurazione indicata.

   - NelConfigurazionescheda, pernome dello stato, inserisci**`Get credit limit`**.

5. DalFlussotab, trascina e rilascia unSceltastato al di sotto delOttieni un limite di creditostato. Rinomina ilSceltada stato a**`Credit applied >= 5000?`**.
6. Trascina e rilascia i seguenti stati come rami diCredito applicato >= 5000?stato.
   a. Pubblicazione su Amazon SNS— DalAzionischeda, trascina e rilascia ilPubblicazione su Amazon SNSAzione API. Rinomina questo stato in**`Wait for human approval`**.
   b. Passarestato: dalFlussoscheda, trascina e rilascia ilPassastato. Rinomina questo ramo in**`Auto-approve limit`**.
7. Trascina e rilascia unPassastato al di sotto delAttendi l'approvazione umanastato. Rinomina questoPassastato a**`Credit limit approved`**.
8. Trascina e rilascia unParallelostato dopo ilSceltadichiarare come segue:
   a. Lascia indicata laParallelostato dopo ilLimite di credito approvatostato.
   b. Rinomina ilParallelostato a**`Verify applicant's identity and address`**.
   c. Sotto entrambi i rami delParallelostato, trascina e rilascia dueAWS LambdaInvocaAzioni API.
   d. Rinomina questi stati come**`Verify identity`**e**`Verify address`**rispettivamente.
   e. Scegli ilLimite di approvazione automaticastato e perStato successivo, puoi lasciare indicata laVerifica l'identità e l'indirizzo del richiedente.
9. Trascina unScansione DynamoDBdichiaralo e inseriscilo sotto ilVerifica l'identità e l'indirizzo del richiedentestato. Rinomina ilScansione DynamoDBda stato a**`Get list of credit bureaus`**.
10. Trascina e rilascia unMappastato dopo ilOttieni un elenco delle agenzie di creditostato. Configura ilMappaindicare come segue:
    a. Rinominalo in**`Get scores from all credit bureaus`**.
    b. PerModalità di elaborazione, puoi lasciare indicata la selezione predefinita diIn linea.
    c. Trascina e rilascia unAWS LambdaInvocaAzione API sullo stato vuoto etichettatoElimina lo stato qui.
    d. Rinomina ilAWS LambdaInvocadichiarare a**`Get all scores`**.
11. Tieni aperta questa finestra e passa al tutorial successivo per ulteriori azioni.

Fasi successive

Nel prossimo tutorial, imparerai come integrare la funzione Lambda utilizzata dalOttieni un limite di creditostato.

Tutorial 2: Definizione della prima integrazione del servizio utilizzando una funzione Lambda

In questo tutorial, imparerai come definire la prima integrazione di servizi per il tuo flusso di lavoro. Si utilizza lo `Task` stato denominato Get credit limit per richiamare una funzione Lambda. All'interno `Task` degli stati, puoi utilizzare le integrazioni AWS SDK supportate da Step Functions.

Per definire la prima integrazione di servizi per il tuo flusso di lavoro, crea innanzitutto una funzione Lambda. Quindi, aggiorna il flusso di lavoro per specificare l'integrazione del servizio con la funzione Lambda. La funzione Lambda utilizzata in questo tutorial restituisce un numero intero generato casualmente che rappresenta il limite di credito richiesto da un richiedente.

Argomenti

- Fase 1: Creare e testare la funzione Lambda
- Passaggio 2: aggiorna il flusso di lavoro: configura lo stato Get credit limit
- Passaggi successivi

Fase 1: Creare e testare la funzione Lambda

Puoi scrivere il codice per la funzione nel AWS Management Console tuo editor preferito. Nei passaggi seguenti, si crea una funzione Lambda Node.js intitolata. `RandomNumberforCredit`

> **Important**
>
> Assicurati che il prototipo di flusso di lavoro che hai creato nel Tutorial 1 sia Regione AWS uguale alla funzione Lambda che creerai in questo tutorial.

1. In una nuova scheda o finestra, apri la console Lambda e crea una funzione Lambda Node.js 16.x intitolata. **`RandomNumberforCredit`** Per informazioni sulla creazione di una funzione

Lambda utilizzando la console, consulta Create a Lambda function in the console nella Developer Guide.AWS Lambda

2. Nella RandomNumberforCreditpagina, scegli index.mjs e sostituisci il codice esistente nell'area del codice sorgente con il codice seguente.

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

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

};
```

3. Dalla sezione Panoramica delle funzioni, copia l'Amazon Resource Name della funzione Lambda e salvalo in un file di testo. Avrai bisogno della funzione ARN mentre specifichi l'integrazione del servizio per lo stato Get credit limit. Di seguito è riportato un esempio di ARN:

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

4. Scegli Deploy, quindi scegli Test per distribuire le modifiche e vedere l'output della funzione Lambda.

Passaggio 2: aggiorna il flusso di lavoro: configura lo stato Get credit limit

Nella console Step Functions, aggiornerai il tuo flusso di lavoro per specificare l'integrazione del servizio con la funzione `RandomNumberforCredit` Lambda che hai creato nella Fase 1.

1. Apri la finestra della console Step Functions contenente il prototipo di workflow creato nel Tutorial 1.
2. Scegli lo stato Ottieni limite di credito e, nella scheda Configurazione, procedi come segue:

   a. Per il tipo di integrazione, mantieni la selezione predefinita di Ottimizzato.

      Utilizzando Step Functions, puoi integrarti con altri Servizi AWS e orchestrarli nei tuoi flussi di lavoro. Per ulteriori informazioni sulle integrazioni di servizi e sui relativi tipi, consulta. Utilizzo AWS Step Functions con altri servizi

   b. Per Nome funzione, scegli la funzione RandomNumberforCreditLambda dall'elenco a discesa.

   c. Mantieni le selezioni predefinite per il resto degli elementi.

3. Tieni aperta questa finestra e passa al tutorial successivo per ulteriori azioni.

> **Note**
>
> In questo tutorial, hai imparato a integrarti con una funzione Lambda all'interno di uno `Task` stato dei tuoi flussi di lavoro. Puoi anche utilizzare altre integrazioni AWS SDK supportate `Task` nello stato specificando il nome del servizio e la chiamata API, come mostrato nella seguente sintassi:
>
> ```
> arn:aws:states:::aws-sdk:serviceName:apiAction
> ```
>
> Per ulteriori informazioni, consulta Utilizzo AWS Step Functions con altri servizi.

Passaggi successivi

Nel prossimo tutorial, implementerai la logica condizionale nel tuo flusso di lavoro. La logica condizionale nelle macchine a stati Step Functions si comporta in modo simile a un'istruzione if-else nei linguaggi di programmazione più comuni. Utilizzerai la logica condizionale nel tuo flusso di lavoro per determinare il percorso di esecuzione in base alle informazioni condizionali.

Tutorial 3: Implementa una condizione if-else nel tuo flusso di lavoro

Puoi implementare le condizioni if-else nei tuoi flussi di lavoro utilizzando lo stato. `Choice` Determina il percorso di esecuzione del flusso di lavoro in base al fatto che una condizione specificata sia vera o falsa.

In questo tutorial, aggiungerai la logica condizionale per determinare se l'importo del credito applicato restituito dalla funzione `RandomNumberforCredit` Lambda utilizzata nel Tutorial 2 supera un limite di soglia specifico. Se la quantità supera il limite di soglia, la domanda richiede l'intervento umano per l'approvazione. In caso contrario, l'applicazione viene approvata automaticamente e passa alla fase successiva.

Simulerai la fase di interazione umana sospendendo l'esecuzione del flusso di lavoro fino alla restituzione di un task token. Per fare ciò, passerai un task token all'integrazione AWS SDK che utilizzerai in questo tutorial, che è Amazon Simple Notification Service. L'esecuzione del flusso

di lavoro verrà sospesa fino a quando non riceverà nuovamente il task token con una chiamata `SendTaskSuccess`API. Per ulteriori informazioni sull'utilizzo dei token di attività, vedereAttendere un callback con il token dell'attività.

Poiché hai già definito i passaggi per l'approvazione umana e l'approvazione automatica nel tuo prototipo di flusso di lavoro, in questo tutorial crei innanzitutto un argomento Amazon SNS che riceve il token di callback. Quindi, si crea una funzione Lambda per implementare la funzionalità di callback. Infine, aggiorni il prototipo del tuo flusso di lavoro aggiungendo i dettagli di queste Servizio AWS integrazioni.

Argomenti

- Fase 1: Crea un argomento Amazon SNS che riceva il token di callback
- Passaggio 2: creare una funzione Lambda per gestire il callback
- Passaggio 3: aggiorna il flusso di lavoro: aggiungi la logica delle condizioni if-else nello stato Choice
- Fasi successive

Fase 1: Crea un argomento Amazon SNS che riceva il token di callback

Per implementare la fase di interazione umana, pubblicherai su un argomento di Amazon Simple Notification Service e passerai il token dell'attività di callback a questo argomento. L'attività di callback interromperà l'esecuzione del flusso di lavoro fino a quando il token dell'attività non verrà restituito con un payload.

1. Apri la console Amazon SNS e crea un tipo di argomento Standard. Per informazioni sulla creazione di un argomento, consulta Creare un argomento Amazon SNS nella Amazon Simple Notification Service Developer Guide.
2. Specifica il nome dell'argomento come**`TaskTokenTopic`**.
3. Assicurati di copiare l'argomento ARN e di salvarlo in un file di testo. Avrai bisogno dell'argomento ARN mentre specifichi l'integrazione del servizio per lo stato di attesa di approvazione umana. Di seguito è riportato un esempio di argomento ARN:

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

4. Crea un abbonamento via e-mail per l'argomento, quindi conferma l'iscrizione. Per informazioni sulla sottoscrizione a un argomento, consulta Creare un abbonamento all'argomento nella Amazon Simple Notification Service Developer Guide.

Passaggio 2: creare una funzione Lambda per gestire il callback

Per gestire la funzionalità di callback, definirai una funzione Lambda e aggiungerai l'argomento Amazon SNS che hai creato nel passaggio 1 come trigger per questa funzione. Quando pubblichi sull'argomento Amazon SNS con un task token, la funzione Lambda viene richiamata con il payload del messaggio pubblicato.

- Fase 2.1: Creare la funzione Lambda per gestire il callback
- Fase 2.2: aggiungi l'argomento Amazon SNS come trigger per la funzione Lambda
- Fase 2.3: Fornire le autorizzazioni necessarie per il ruolo IAM della funzione Lambda

Fase 2.1: Creare la funzione Lambda per gestire il callback

In questa funzione, elaborerai la richiesta di approvazione del limite di credito e restituirai il risultato della richiesta con esito positivo con la chiamata `SendTaskSuccess`API. Questa funzione Lambda restituirà anche il task token ricevuto dall'argomento Amazon SNS.

Per semplicità, la funzione Lambda utilizzata per la fase di interazione umana approva automaticamente qualsiasi attività e restituisce il token dell'operazione con una chiamata `SendTaskSuccess` API. È possibile denominare la funzione Lambda come**`callback-human-approval`**.

1. In una nuova scheda o finestra, apri la console Lambda e crea una funzione Lambda Node.js 16.x intitolata. **`callback-human-approval`** Per informazioni sulla creazione di una funzione Lambda utilizzando la console, vedi Creare una funzione Lambda nella console nella Guida per gli AWS Lambdasviluppatori.
2. Nella callback-human-approvalpagina, sostituisci il codice esistente nell'area Codice sorgente con il codice seguente.

```
// 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. Tieni aperta questa finestra ed esegui i passaggi indicati nella sezione successiva per ulteriori azioni.

Fase 2.2: aggiungi l'argomento Amazon SNS come trigger per la funzione Lambda

Quando aggiungi l'argomento Amazon SNS che hai creato nella fase 1 di questo tutorial come trigger per la funzione Lambda creata nella fase 2.1 di questo tutorial, la funzione Lambda viene attivata ogni volta che pubblichi sull'argomento Amazon SNS. Quando pubblichi sull'argomento Amazon SNS con un task token, la funzione Lambda viene richiamata con il payload del messaggio pubblicato. Per ulteriori informazioni sulla configurazione dei trigger per le funzioni Lambda, consulta Configurazione dei trigger nella Guida per gli sviluppatori. AWS Lambda

1. Nella sezione Panoramica delle funzioni della funzione `callback-human-approval` Lambda, scegli Aggiungi trigger.
2. Dall'elenco a discesa dei trigger, scegli SNS come trigger.
3. Per l'argomento SNS, inizia a digitare il nome dell'argomento Amazon SNS che hai creato nella fase 1 di questo tutorial e sceglilo dall'elenco a discesa visualizzato.
4. Scegli Add (Aggiungi).

5. Tieni aperta questa finestra ed esegui i passaggi indicati nella sezione successiva per ulteriori azioni.

Fase 2.3: Fornire le autorizzazioni necessarie per il ruolo IAM della funzione Lambda

È necessario fornire alla funzione `callback-human-approval` Lambda le autorizzazioni per accedere a Step Functions per restituire il task token insieme alla chiamata `SendTaskSucess` API.

1. Nella callback-human-approvalpagina, scegli la scheda Configurazione, quindi scegli Autorizzazioni.
2. In Ruolo di esecuzione, scegli il nome del ruolo per accedere alla pagina Ruoli della AWS Identity and Access Management console.
3. Per aggiungere l'autorizzazione richiesta, scegli Aggiungi autorizzazioni, quindi scegli Allega criteri.
4. Nella casella di ricerca, digita **`AWSStepFunctions`** e premi Invio.
5. Scegli, AWSStepFunctionsFullAccessquindi scorri verso il basso per selezionare Allega politiche. Questo aggiunge la politica contenente l'autorizzazione necessaria per il ruolo della funzione `callback-human-approval` Lambda.

Passaggio 3: aggiorna il flusso di lavoro: aggiungi la logica delle condizioni if-else nello stato Choice

Nella console Step Functions, definisci la logica condizionale per il tuo flusso di lavoro utilizzando lo `Choice` stato. Se l'output restituito dalla funzione `RandomNumberforCredit` Lambda è inferiore a 5000, il credito richiesto viene approvato automaticamente. Se l'output restituito è maggiore o uguale a 5000, l'esecuzione del flusso di lavoro procede alla fase di interazione umana per l'approvazione del limite di credito.

Nello `Choice` stato, si utilizza un operatore di confronto per confrontare una variabile di input con un valore specifico. È possibile specificare la variabile di input come input di esecuzione durante l'avvio dell'esecuzione di una macchina a stati o utilizzare l'output di un passaggio precedente come input per il passaggio corrente. Per impostazione predefinita, l'output di un passaggio viene memorizzato in una variabile chiamata`Payload`. Per utilizzare il valore della `Payload` variabile per il confronto nello `Choice` stato, utilizzate la $ sintassi illustrata nella procedura seguente.

Per informazioni su come le informazioni fluiscono da uno stato all'altro e su come specificare input e output nei flussi di lavoro, vedere Tutorial 7: Configurazione di input e output e. Elaborazione di input e output in Step Functions

> **Note**
>
> Se lo `Choice` stato utilizza una variabile di input specificata nell'input di esecuzione della macchina a stati per il confronto, utilizza la `$.variable_name` sintassi per eseguire il confronto. Ad esempio, per confrontare una variabile, ad esempio`myAge`, usa la sintassi`$.myAge`.

Poiché in questo passaggio, lo `Choice` stato riceverà input dallo stato Ottieni limite di credito, utilizzerai la $ sintassi per la configurazione dello `Choice` stato. Per scoprire in che modo il risultato dell'esecuzione della macchina a stati differisce quando si utilizza la `$.variable_name` sintassi nella configurazione dello `Choice` stato per fare riferimento all'output di un passaggio precedente, vedere la Effettuare il debug del percorso non valido Errore di stato della scelta sezione del Tutorial 8.

Per aggiungere la logica delle condizioni if-else utilizzando lo stato **`Choice`**

1. Apri la finestra della console Step Functions contenente il prototipo del flusso di lavoro in cui hai creato. Tutorial 1: Crea il prototipo per la tua macchina a stati
2. Scegli il credito applicato >= 5000? state e nella scheda Configurazione, specificare la logica condizionale come segue:
   a. In Regole di scelta, scegli l'icona Modifica nel riquadro Regola #1 per definire la regola di prima scelta.
   b. Scegli Aggiungi condizioni.
   c. Nella finestra di dialogo Condizioni per la regola #1, per Variabile, immettere**$**.
   d. Per Operator, la scelta è inferiore a.
   e. Per Valore, scegli Numero costante, quindi inserisci il **`5000`** campo accanto all'elenco a discesa Valore.
   f. Scegli Salva condizioni.
   g. Per lo stato Quindi successivo è: elenco a discesa, scegli Limite di approvazione automatica.

h. Scegliete Aggiungi nuova regola di scelta, quindi definite la regola della seconda scelta quando l'importo del credito è maggiore o uguale a 5000 ripetendo i passaggi da 2.b a 2.f. Per Operator, la scelta è maggiore o uguale a.

i. Per lo stato Quindi successivo è: elenco a discesa, scegli Attendi l'approvazione umana.

j. Nella casella Regola predefinita, scegli l'icona Modifica per definire la regola di scelta predefinita, quindi scegli Attendi l'approvazione umana dall'elenco a discesa Stato predefinito. Definisci la regola Default per specificare lo stato successivo a cui passare se nessuna delle condizioni dello stato di scelta restituisce vero o falso.

3. Configura lo stato di attesa dell'approvazione umana come segue:

a. Nella scheda Configurazione, per Argomento, inizia a digitare il nome dell'argomento Amazon SNS e scegli il nome visualizzato nell'elenco a discesa. TaskTokenTopic

b. Per Messaggio, scegli Inserisci messaggio dall'elenco a discesa. Nel campo Messaggio, specifichi il messaggio che desideri pubblicare sull'argomento di Amazon SNS. Per questo tutorial, pubblichi un task token come messaggio.

   Un task token consente di sospendere un flusso di lavoro Step Functions di tipo standard fino al completamento di un processo esterno e alla restituzione del task token. Quando si specifica uno stato Task come attività di callback specificando il modello di integrazione del `.waitForTaskToken` servizio, viene generato un task token che viene inserito nell'oggetto di contesto all'avvio dell'attività. L'oggetto di contesto è una struttura JSON interna disponibile durante un'esecuzione e contiene informazioni sulla macchina a stati e sulla sua esecuzione. Per ulteriori informazioni sugli oggetti di contesto, vedereOggetto Context.

c. Nella casella visualizzata, inserisci quanto segue come messaggio:

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

d. Seleziona la casella di controllo Attendi la richiamata.

e. Scegli Fine nella finestra di dialogo che appare.

4. Tieni aperta questa finestra e passa al tutorial successivo per ulteriori azioni.

Fasi successive

Nel prossimo tutorial, imparerai come eseguire più attività in parallelo.

Tutorial 4: Definisci più attività da eseguire in parallelo

Finora hai imparato a eseguire i flussi di lavoro in modo sequenziale. Tuttavia, è possibile eseguire due o più passaggi in parallelo utilizzando lo `Parallel` stato. Uno `Parallel` stato fa sì che l'interprete esegua ogni ramo contemporaneamente.

Entrambi i rami di uno `Parallel` stato ricevono lo stesso input, ma ogni ramo elabora le parti di input specifiche per esso. Step Functions attende il completamento dell'esecuzione di ogni ramo prima di procedere al passaggio successivo.

In questo tutorial, si utilizza lo stato parallelo per verificare contemporaneamente l'identità e l'indirizzo del richiedente.

Argomenti

- Fase 1: Creare le funzioni Lambda per eseguire i controlli richiesti
- Fase 2: Aggiornamento del flusso di lavoro: aggiungere attività parallele da eseguire

Fase 1: Creare le funzioni Lambda per eseguire i controlli richiesti

Questo flusso di lavoro relativo all'applicazione della carta di credito richiama due funzioni Lambda nello stato parallelo per verificare l'identità e l'indirizzo del richiedente. Questi controlli vengono eseguiti contemporaneamente utilizzando lo stato Parallel. La macchina a stati completa l'esecuzione solo dopo che entrambi i rami paralleli hanno completato l'esecuzione.

Per creare le funzioni Lambda check-identity e check-address

1. In una nuova scheda o finestra, apri la console Lambda e crea due funzioni Lambda di Node.js 16.x intitolate e. `check-identity` `check-address` Per informazioni sulla creazione di una funzione Lambda utilizzando la console, vedi Creare una funzione Lambda nella console nella Guida per gli AWS Lambdasviluppatori.
2. Apri la pagina della funzione check-identity e sostituisci il codice esistente nell'area Codice sorgente con il seguente codice:

```
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. Apri la pagina della funzione di controllo dell'indirizzo e sostituisci il codice esistente nell'area Codice sorgente con il seguente codice:

```
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. Per entrambe le funzioni Lambda, dalla sezione Panoramica delle funzioni, copia i rispettivi Amazon Resource Names (ARN) e salvali in un file di testo. Avrai bisogno della funzione ARNs mentre specifichi l'integrazione del servizio per l'identità e lo stato dell'indirizzo del richiedente Verify. Di seguito è riportato un esempio di ARN:

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

Fase 2: Aggiornamento del flusso di lavoro: aggiungere attività parallele da eseguire

Nella console Step Functions, aggiornerai il tuo flusso di lavoro per specificare l'integrazione del servizio con le funzioni Lambda di check-identity e check-address che hai creato nel passaggio 1.

Per aggiungere attività parallele nel flusso di lavoro

1. Apri la finestra della console Step Functions contenente il prototipo del flusso di lavoro in cui hai creato. Tutorial 1: Crea il prototipo per la tua macchina a stati
2. Scegli Verifica lo stato dell'identità e, nella scheda Configurazione, procedi come segue:

a. Per Tipo di integrazione, mantieni la selezione predefinita di Ottimizzato.

> **Note**
>
> Utilizzando Step Functions, puoi integrarli con altri Servizi AWS e orchestrarli nei tuoi flussi di lavoro. Per ulteriori informazioni sulle integrazioni dei servizi e sui relativi tipi, vedere Utilizzo AWS Step Functions con altri servizi

b. Per Nome funzione, scegli la funzione checkidentity Lambda dall'elenco a discesa.

c. Per Payload, scegli Inserisci payload, quindi sostituisci il payload di esempio con quanto segue come payload:

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

3. Scegli lo stato di verifica dell'indirizzo e, nella scheda Configurazione, procedi come segue:

a. Per Tipo di integrazione, mantieni la selezione predefinita di Ottimizzato.

b. Per Nome funzione, scegli la funzione Lambda di controllo dell'indirizzo dall'elenco a discesa.

c. Per Payload, scegli Inserisci payload, quindi sostituisci il payload di esempio con quanto segue come payload:

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

4. Seleziona Successivo.

Tutorial 5: Iterazione simultanea su una raccolta di elementi

Nel tutorial precedente, hai imparato a eseguire rami separati di passaggi in parallelo usando lo `Parallel` stato. Utilizzando lo `Map` stato, è possibile eseguire una serie di passaggi del flusso di

lavoro per ogni elemento di un set di dati. Le iterazioni `Map` dello stato vengono eseguite in parallelo, il che rende possibile elaborare rapidamente un set di dati.

Includendo lo `Map` stato nei flussi di lavoro è possibile eseguire attività, come l'elaborazione dei dati, utilizzando una delle due modalitàModalità di elaborazione dello stato della mappa: modalità in linea e modalità distribuita. Per configurare uno `Map` stato, si definisce uno`ItemProcessor`, che contiene oggetti JSON che specificano la modalità di elaborazione dello `Map` stato e la relativa definizione. In questo tutorial, esegui lo `Map` stato nella modalità Inline predefinita, che supporta fino a 40 iterazioni simultanee. Quando si esegue lo `Map` stato in modalità Distribuita, supporta fino a 10.000 esecuzioni parallele di flussi di lavoro secondari.

Quando l'esecuzione del flusso di lavoro entra nello `Map` stato, eseguirà un'iterazione su un array JSON specificato nell'input dello stato. Per ogni elemento dell'array, l'iterazione corrispondente viene eseguita nel contesto del flusso di lavoro che contiene lo stato. `Map` Quando tutte le iterazioni sono complete, lo `Map` stato restituirà un array contenente l'output per ogni elemento elaborato da. `ItemProcessor`

In questo tutorial, imparerai come utilizzare lo `Map` stato in modalità Inline per recuperare il punteggio di credito di un richiedente iterando su una serie di agenzie di credito. A tale scopo, devi prima recuperare i nomi di tutte le agenzie di credito archiviate in una tabella Amazon DynamoDB e quindi utilizzare lo stato per scorrere l'elenco `Map` delle agenzie di credito per recuperare il punteggio di credito del richiedente riportato da ciascuna di queste agenzie.

Argomenti

- Fase 1: Creare una tabella DynamoDB per memorizzare il nome di tutte le agenzie di credito
- Fase 2: Aggiornare la macchina a stati — Recupera i risultati dalla tabella DynamoDB
- Fase 3: Creare una funzione Lambda che restituisca i punteggi di credito per tutte le agenzie di credito
- Passaggio 4: Aggiorna la macchina a stati: aggiungi uno stato della mappa per recuperare in modo iterativo i punteggi di credito

Fase 1: Creare una tabella DynamoDB per memorizzare il nome di tutte le agenzie di credito

In questo passaggio, si crea una tabella denominata **`GetCreditBureau`** utilizzando la console DynamoDB. La tabella utilizza l'attributo stringa Name come chiave di partizione. In questa tabella,

memorizzi il nome di tutte le agenzie di credito da cui desideri recuperare il punteggio di credito del richiedente.

1. Accedi AWS Management Console e apri la console DynamoDB all'indirizzo https://console.aws.amazon.com/dynamodb/.
2. Nel pannello di navigazione della console, scegli Tabelle, quindi scegli Crea tabella.
3. Inserisci i dettagli della tabella come segue:
   a. Nel campo Table name (Nome tabella) immetti **`GetCreditBureau`**.
   b. Per Partition key (Chiave di partizione), inserisci **`Name`**.
   c. Mantieni le selezioni predefinite e scegli Crea tabella.
4. Dopo aver creato la tabella, nell'elenco Tabelle, scegli la GetCreditBureautabella.
5. Scegli Azioni, quindi scegli Crea elemento.
6. In Valore, inserisci il nome di un ufficio di credito. Ad esempio, **`CredTrack`**.
7. Scegli Create Item (Crea elemento).
8. Ripeti questo processo e crea articoli per i nomi di altre agenzie di credito. Ad esempio **`KapFinn`** e **`CapTrust`**.

Fase 2: Aggiornare la macchina a stati — Recupera i risultati dalla tabella DynamoDB

Nella console Step Functions, aggiungerai uno `Task` stato e utilizzerai l'integrazione AWS SDK per recuperare i nomi delle agenzie di credito dalla tabella DynamoDB che hai creato nello Step 1. Utilizzerai l'output di questo passaggio come input per `Map` lo stato che aggiungerai successivamente al flusso di lavoro in questo tutorial.

1. Apri la macchina a CreditCardWorkflowstati per aggiornarla.
2. Scegli lo stato Ottieni l'elenco delle agenzie di credito.
3. Per i parametri API, specifica il valore del nome della tabella come**`GetCreditBureau`**.

Fase 3: Creare una funzione Lambda che restituisca i punteggi di credito per tutte le agenzie di credito

In questo passaggio, crei una funzione Lambda che riceve i nomi di tutte le agenzie di credito come input e restituisce il punteggio di credito del richiedente per ciascuna di queste agenzie di credito. Questa funzione Lambda verrà richiamata dallo `Map` stato che aggiungerai al tuo flusso di lavoro nel passaggio 4 di questo tutorial.

1. Creare una funzione Lambda Node.js 16.x e assegnarle un nome. **`get-credit-score`**
2. Nella pagina intitolata get-credit-score, incolla il codice seguente nell'area Codice sorgente.

```
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. Distribuire la funzione Lambda.

Passaggio 4: Aggiorna la macchina a stati: aggiungi uno stato della mappa per recuperare in modo iterativo i punteggi di credito

Nella console Step Functions, aggiungi uno `Map` stato che richiama la funzione get-credit-scoreLambda per controllare il punteggio di credito del richiedente per tutte le agenzie di credito restituite dallo stato Get list of credit bureaus.

1. Apri la macchina a CreditCardWorkflowstati per aggiornarla.
2. Scegli lo stato Ottieni punteggi da tutte le agenzie di credito.

3. Nella scheda Configurazione, scegli Fornisci un percorso per l'array degli elementi e quindi inserisci**`$.Items`**.
4. Scegli Ottieni tutti i punteggi all'interno dello `Map` stato.
5. Nella scheda Configurazione, assicurati che per Tipo di integrazione sia selezionato Optimized.
6. Per Nome funzione, inizia a digitare il nome della funzione get-credit-scoreLambda e selezionalo dall'elenco a discesa visualizzato.
7. Per Payload, scegli Nessun payload.

Tutorial 6: Salvare il flusso di lavoro ed eseguire la macchina a stati

Ora che hai configurato le risorse di tutto ciò che Servizi AWS stai utilizzando nel prototipo del flusso di lavoro, puoi salvarlo come macchina a stati Step Functions e iniziare a eseguirlo.

Argomenti

- Passaggio 1: rivedere la definizione della macchina a stati generata automaticamente e salvare la macchina a stati
- Passaggio 2: aggiungi le politiche IAM rimanenti
- Fase 3: Esegui la macchina a stati

Passaggio 1: rivedere la definizione della macchina a stati generata automaticamente e salvare la macchina a stati

Mentre trascini gli stati dalla scheda Flow sull'area di disegno di Workflow Studio per creare il prototipo del flusso di lavoro, Step Functions compone automaticamente la definizione Amazon States Language (ASL) del flusso di lavoro in tempo reale. È possibile modificare questa definizione come richiesto in. Editor del codice

Per esaminare la definizione ASL e salvare la macchina a stati

1. (Facoltativo) Scegliete Definizione su ✓Inspector per visualizzare la definizione della macchina a stati Amazon States Language (ASL), che viene generata automaticamente in base alle selezioni effettuate nelle schede Azioni e Flusso e nel pannello Inspector.

> **Tip**
>
> Per modificare la definizione, puoi aprire l'editor di codice selezionando Codice nella parte superiore della pagina. Per questo tutorial, continua con la definizione generata automaticamente.

2. Specificate un nome per la vostra macchina a stati. Per fare ciò, scegli l'icona di modifica accanto al nome della macchina a stati predefinita di MyStateMachine. Quindi, nella configurazione della macchina a stati, specifica un nome nella casella Nome macchina a stati.

   Per questo tutorial, inserisci il nome **`CreditCardWorkflow`**.

3. (Facoltativo) Nella configurazione della macchina a stati, specificate altre impostazioni del flusso di lavoro, come il tipo di macchina a stati e il relativo ruolo di esecuzione.

   Per questo tutorial, mantieni tutte le selezioni predefinite nelle impostazioni della macchina a stati.

   > **Note**
   >
   > (Facoltativo) Step Functions crea automaticamente un ruolo di esecuzione per la macchina a stati con i privilegi minimi richiesti per richiamare la funzione `RandomNumberforCredit` Lambda e pubblicarla sull'argomento Amazon SNS.

   Se in precedenza hai creato un ruolo IAM con le autorizzazioni corrette per la tua macchina a stati e desideri utilizzarlo, in Autorizzazioni seleziona Scegli un ruolo esistente, quindi seleziona un ruolo dall'elenco. Oppure seleziona Inserisci un ruolo ARN e quindi fornisci un ARN per quel ruolo IAM.

4. Nella finestra di dialogo Conferma la creazione del ruolo, scegli Conferma per continuare.

   Puoi anche scegliere Visualizza le impostazioni del ruolo per tornare alla configurazione della macchina a stati.

   > **Note**
   >
   > Se elimini il ruolo IAM creato da Step Functions, Step Functions non può ricrearlo in un secondo momento. Allo stesso modo, se modifichi il ruolo (ad esempio, rimuovendo

Step Functions dai principi nella policy IAM), Step Functions non può ripristinare le impostazioni originali in un secondo momento.

Passaggio 2: aggiungi le politiche IAM rimanenti

Poiché Step Functions non genera automaticamente le autorizzazioni per richiamare le funzioni Lambda utilizzate nello `Parallel` stato, è necessario aggiungere la policy necessaria.

Per aggiungere la politica rimanente

1. Nella CreditCardWorkflowpagina, scegli il ruolo IAM per la tua macchina a stati per accedere alla console IAM. In questa pagina aggiungerai le autorizzazioni necessarie per le restanti funzioni Lambda.
2. Seleziona Aggiungi autorizzazioni, quindi seleziona Collega policy.
3. Nella casella di ricerca, digita **`AWSLambdaRole`** e premi Invio.
4. Scegli AWSLambdaRolee quindi seleziona Allega politiche. Questa politica viene ora aggiunta al ruolo di esecuzione della tua macchina a stati. Questo criterio consente di richiamare qualsiasi funzione Lambda nella macchina a stati.

Fase 3: Esegui la macchina a stati

Le esecuzioni di macchine a stati sono casi in cui si esegue il flusso di lavoro per eseguire attività.

Per eseguire la macchina a stati

1. Nella CreditCardWorkflowpagina, scegli Avvia esecuzione.

   Viene visualizzata la finestra di dialogo Avvia esecuzione.
2. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

   a. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

> **Note**
>
> Step Functions consente di creare nomi per macchine a stati, esecuzioni, attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

> **Note**
>
> Non è necessario fornire alcun input per eseguire questa macchina a stati. Tuttavia, è possibile specificare un input di esecuzione, se necessario, nell'area Input della finestra di dialogo Avvia esecuzione per altre macchine a stati. Per un esempio di come fornire l'input di esecuzione a una macchina a stati, vedete Fase 4: Avvio di una nuova esecuzione del tutorial Impara a usare il AWS Step Functions Workflow Studio.

b. Selezionare Start execution (Avvia esecuzione).

3. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

   Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Tutorial 7: Configurazione di input e output

Un'esecuzione di Step Functions riceve un testo JSON come input e passa tale input al primo stato del flusso di lavoro. I singoli stati di un flusso di lavoro ricevono i dati JSON come input e di solito passano i dati JSON come output allo stato successivo. Per impostazione predefinita, i dati passano da uno stato allo stato successivo del flusso di lavoro a meno che non sia stato configurato l'input e/o l'output per uno o più stati del flusso di lavoro. Comprendere come le informazioni fluiscono

da uno stato all'altro e imparare a filtrare e manipolare questi dati è fondamentale per progettare e implementare in modo efficace i flussi di lavoro in Step Functions.

Step Functions fornisce più filtri per controllare il flusso di dati di input e output tra gli stati. I seguenti filtri sono disponibili per l'uso nei tuoi flussi di lavoro:

> **Note**
>
> In base al tuo caso d'uso, potresti non aver bisogno di applicare tutti questi filtri nei tuoi flussi di lavoro.

`InputPath`

Seleziona QUALE parte dell'intero payload di input da utilizzare come input di un'attività. Se si specifica questo campo, Step Functions applica innanzitutto questo campo.

`Parametri`

Specifica COME deve apparire l'input prima di richiamare l'operazione. Con il `Parameters` campo, puoi creare una raccolta di coppie chiave-valore che vengono passate come input a un'Servizio AWSintegrazione, ad esempio una AWS Lambda funzione. Questi valori possono essere statici o selezionati dinamicamente dall'input di stato o dall'oggetto di contesto del flusso di lavoro.

`ResultSelector`

Determina COSA scegliere dall'output di un'attività. Con il `ResultSelector` campo, puoi creare una raccolta di coppie chiave-valore che sostituiscono il risultato di uno stato e passano quella raccolta a. `ResultPath`

`ResultPath`

Determina DOVE inserire l'output di un'attività. Utilizzate il `ResultPath` per determinare se l'output di uno stato è una copia del suo input, il risultato che produce o una combinazione di entrambi.

`OutputPath`

Determina COSA inviare allo stato successivo. Con`OutputPath`, puoi filtrare le informazioni indesiderate e trasmettere solo la parte di dati JSON che ti interessa.

> **Tip**
>
> `ResultSelector`I filtri `Parameters` and funzionano costruendo JSON, mentre i `OutputPath` filtri `InputPath` and funzionano filtrando nodi specifici all'interno di un oggetto dati JSON e il `ResultPath` filtro funziona creando un campo in cui è possibile aggiungere l'output.

In questo tutorial, imparerai a eseguire le seguenti attività:

- Seleziona porzioni specifiche dell'input grezzo utilizzando il InputPath filtro
- Manipola l'input selezionato utilizzando il filtro Parametri
- Configurare l'output utilizzando OutputPath i filtri ResultSelectorResultPath, e

Per ulteriori informazioni sulla configurazione di input e output nei flussi di lavoro, consulta. Elaborazione di input e output in Step Functions

Seleziona porzioni specifiche dell'input grezzo utilizzando il InputPath filtro

Utilizzate il `InputPath` filtro per selezionare una parte specifica del payload di input.

Se non lo specificate`InputPath`, il suo valore predefinito è$, il che fa sì che l'attività dello stato faccia riferimento all'intero input non elaborato anziché a una parte specifica.

Per informazioni su come utilizzare il InputPath filtro, effettuate le seguenti operazioni:

- Passaggio 1: creare una macchina a stati
- Passaggio 2: Esegui la macchina a stati
- Passaggio 3: utilizzare il InputPath filtro per selezionare parti specifiche di un input di esecuzione

Passaggio 1: creare una macchina a stati

> **Important**
>
> Assicurati che la tua macchina a stati utilizzi lo stesso AWS account e la stessa regione della funzione Lambda che hai creato in precedenza.

1. Usa l'esempio di `Parallel` stato che hai appreso nel Tutorial 4 per creare una nuova macchina a stati. Assicurati che il prototipo del tuo flusso di lavoro sia simile al prototipo seguente.
2. Configura le integrazioni per le funzioni `check-identity` e `check-address` Lambda. Per informazioni sulla creazione delle funzioni Lambda e sul loro utilizzo nella macchina a stati, consulta Fase 1: Creare le funzioni Lambda per eseguire i controlli richiesti e. Fase 2: Aggiornamento del flusso di lavoro: aggiungere attività parallele da eseguire
3. Per Payload, assicurati di mantenere la selezione predefinita di Usa l'input di stato come payload.
4. Scegli Avanti, quindi esegui i passaggi da 1 a 3 Fase 1: Salvare la macchina a stati del Tutorial 5 per creare una nuova macchina a stati. Per questo tutorial, dai un nome alla tua macchina a stati**`WorkflowInputOutput`**.

Passaggio 2: Esegui la macchina a stati

1. Nella WorkflowInputOutputpagina, scegli Avvia esecuzione.
2. (Facoltativo) Per identificare l'esecuzione, puoi specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

> **Note**
>
> Step Functions consente di creare nomi per macchine a stati, esecuzioni, attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

3. Nell'area Input, aggiungi i seguenti dati JSON come input di esecuzione.

```
{
  "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. Selezionare Start execution (Avvia esecuzione).
5. L'esecuzione della macchina a stati genera un errore perché non hai specificato quali parti dell'input di esecuzione devono utilizzare le funzioni `check-identity` e `check-address` Lambda per eseguire la verifica dell'identità e dell'indirizzo richiesta.
6. Continua con la Fase 3 di questo tutorial per correggere l'errore.

Passaggio 3: utilizzare il **`InputPath`** filtro per selezionare parti specifiche di un input di esecuzione

1. Nella pagina Dettagli di esecuzione, scegli Modifica macchina a stati.
2. Per verificare l'identità del richiedente, come indicato nell'input di esecuzione fornito inPassaggio 2: Esegui la macchina a stati, modifica la definizione dell'attività di verifica dell'identità come segue:

```
...
{
    "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
      }
    }
  }
  ...
```

Di conseguenza, i seguenti dati JSON diventano disponibili come input per la `check-identity` funzione.

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

3. Per verificare l'indirizzo del richiedente come indicato nell'input di esecuzione, modifica la definizione dell'`Verify address`attività come segue:

```
...
{
    "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
      }
    }
  }
  ...
```

Di conseguenza, i seguenti dati JSON diventano disponibili come input per la `check-address` funzione.

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

4. Selezionare Start execution (Avvia esecuzione). L'esecuzione della macchina a stati ora viene completata correttamente.

Manipola l'input selezionato utilizzando il filtro Parametri

Sebbene il `InputPath` filtro ti aiuti a limitare l'input JSON non elaborato che fornisci, utilizzando il `Parameters` filtro puoi passare una raccolta di coppie chiave-valore come input. Queste coppie chiave-valore possono essere valori statici definiti nella definizione della macchina a stati o valori selezionati dall'input non elaborato utilizzando. `InputPath`

Nei tuoi flussi di lavoro, `Parameters` vengono applicati dopo`InputPath`. `Parameters`consentono di specificare in che modo l'attività sottostante accetta il payload in ingresso. Ad esempio, se la funzione `check-address` Lambda accetta un parametro stringa come input anziché i dati JSON, puoi utilizzare il `Parameters` filtro per trasformare l'input.

Nell'esempio seguente, il `Parameters` filtro riceve l'input selezionato utilizzando `InputPath` in Passaggio 3: utilizzare il `InputPath` filtro per selezionare parti specifiche di un input di esecuzione e applica la funzione `States.Format` intrinseca agli elementi di input per creare una stringa chiamata. `addressString` Le funzioni intrinseche consentono di eseguire operazioni di elaborazione dei dati di base su un determinato input. Per ulteriori informazioni, consulta Funzioni intrinseche.

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

Di conseguenza, la seguente stringa viene creata e fornita alla funzione `check-address` Lambda come input.

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

Configurare l'output utilizzando OutputPath i filtri ResultSelectorResultPath, e

Quando la funzione `check-address` Lambda viene richiamata nella macchina a WorkflowInputOutputstati, la funzione restituisce un payload di output dopo aver eseguito la verifica

dell'indirizzo. Nella pagina Dettagli dell'esecuzione, scegli la fase Verifica indirizzo e visualizza il payload di output all'interno del risultato dell'attività nel Dettagli del passaggio riquadro.

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

Usando ResultSelector

Ora, se devi fornire il risultato dei controlli di verifica dell'identità e dell'indirizzo ai seguenti stati del tuo flusso di lavoro, puoi selezionare il nodo payload.body nel JSON di output e utilizzare la funzione `StringToJson` intrinseca nel filtro per formattare i dati come richiesto. `ResultSelector`

`ResultSelector`seleziona ciò che è necessario dall'output dell'attività. Nell'esempio seguente, **`ResultSelector`** prende la stringa in $.payload.body e applica la funzione **`States.StringToJson`** intrinseca per convertire la stringa in JSON e inserisce il JSON risultante all'interno del nodo di identità.

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

Di conseguenza, vengono creati i seguenti dati JSON.

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

```
    }
  }
```

Mentre lavori con questi filtri di input e output, puoi anche riscontrare errori di runtime derivanti dalla specificazione di espressioni di percorso JSON non valide. Per ulteriori informazioni, consulta .

Usando ResultPath

È possibile specificare una posizione nel payload di input iniziale per salvare il risultato dell'elaborazione dell'attività di uno stato utilizzando il `ResultPath` campo. Se non lo specifichi`ResultPath`, il suo valore predefinito è$, il che fa sì che il payload di input iniziale venga sostituito con il risultato non elaborato dell'attività. Se si specifica `ResultPath` as`null`, il risultato grezzo viene scartato e il carico utile iniziale in ingresso diventa l'output effettivo.

Se applichi il `ResultPath` campo ai dati JSON creati utilizzando il `ResultSelector` campo, il risultato dell'attività viene aggiunto all'interno del nodo dei risultati nel payload di input, come mostrato nell'esempio seguente:

```
{
  "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
    }
  }
}
```

Usando OutputPath

È possibile selezionare una parte dello stato di output dopo l'applicazione di `ResultPath` per passare allo stato successivo. In questo modo sarà possibile filtrare le informazioni indesiderate e trasferire solo la parte di JSON richiesta.

Nell'esempio seguente, il `OutputPath` campo salva l'output dello stato all'interno del nodo dei risultati:`"OutputPath": "$.results"`. Di conseguenza, l'output finale dello stato, che è possibile passare allo stato successivo, è il seguente:

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

Utilizzo delle funzionalità della console per visualizzare i flussi di dati di input e output

Puoi visualizzare il flusso di dati di input e output tra gli stati nei tuoi flussi di lavoro utilizzando il simulatore di flusso di dati della console Step Functions o l'opzione di visualizzazione avanzata nella pagina Dettagli dell'esecuzione.

Tutorial 8: errori di debug nella console

Lavorando con Step Functions, è possibile che si verifichino errori di runtime dovuti a motivi quali:

- Un percorso JSON non valido per il campo Variabile nello `Choice` stato.
- Problema di definizione della macchina a stati, ad esempio nessuna regola di corrispondenza definita per uno `Choice` stato.
- Espressioni di percorso JSON non valide durante l'applicazione di filtri per manipolare input e output.
- Errori delle attività a causa di un'eccezione della funzione Lambda.
- Errori di autorizzazione IAM.

In questo tutorial, scoprirai come eseguire il debug di alcuni di questi errori utilizzando la console Step Functions. Per ulteriori informazioni, consulta Gestione degli errori in Step Functions.

Argomenti

- Effettuare il debug del percorso non valido Errore di stato della scelta
- Eseguire il debug degli errori di espressione del percorso JSON durante l'applicazione di filtri di input e output

Effettuare il debug del percorso non valido Errore di stato della scelta

Quando specifichi un percorso JSON errato o irrisolvibile nel campo Variabile dello `Choice` stato o non definisci una regola corrispondente nello `Choice` stato, ricevi un errore durante l'esecuzione del flusso di lavoro.

Per illustrare l'errore di percorso non valido, questo tutorial introduce un errore di `Choice` stato nel flusso di lavoro. Utilizzerai la macchina a CreditCardWorkflowstati e ne modificherai la definizione per introdurre l'errore.

1. Apri la console Step Functions e scegli la macchina a CreditCardWorkflowstati.
2. Scegliete Modifica per modificare la definizione della macchina a stati. Apporta la modifica evidenziata nel codice seguente alla definizione della tua macchina a stati.

```
{
  "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. Scegli Salva, quindi scegli Salva comunque.
4. Esegui la macchina statale.
5. Nella pagina Dettagli di esecuzione dell'esecuzione della macchina a stati, esegui una delle seguenti operazioni:

   a. Scegliete Causa nel messaggio di errore per visualizzare il motivo dell'errore di esecuzione.

   b. Scegli Mostra i dettagli della fase nel messaggio di errore per visualizzare la fase che ha causato l'errore.
6. Nella scheda Input & Output della sezione Dettagli della fase, scegli il pulsante di attivazione Visualizzazione avanzata per visualizzare il percorso di trasferimento dei dati di input e output per uno stato selezionato.
7. Nella visualizzazione Grafica, assicurati che il credito sia stato applicato **>=** 5000? viene selezionato ed effettua le seguenti operazioni:

   a. Visualizza il valore di input dello stato nella casella Input.

   b. Scegliete la scheda Definizione e osservate il percorso JSON specificato per il campo Variabile.

      Il valore di input per il credito applicato **>=** 5000? state è un valore numerico, mentre hai specificato il percorso JSON per il valore di input come. `$.Payload` Durante l'esecuzione della macchina a stati, lo `Choice` stato non può risolvere questo percorso JSON perché non esiste.
8. Modifica la macchina a stati per specificare il valore del campo Variabile come**$**.

```
{
  "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"
    },
    ...
    ...
  }
}
```

Eseguire il debug degli errori di espressione del percorso JSON durante l'applicazione di filtri di input e output

Mentre lavori con i filtri di input e output, potresti riscontrare errori di runtime dovuti alla specificazione di espressioni di percorso JSON non valide.

L'esempio seguente utilizza la WorkflowInputOutputmacchina a stati creata nel Tutorial 5 e mostra uno scenario in cui si utilizza il `ResultSelector` filtro per selezionare parti dell'output dell'attività.

1. Applica il `ResultSelector` filtro per scegliere una parte dell'output dell'attività per la fase Verifica identità. Per fare ciò, modifica la definizione della tua macchina a stati come segue:

```
{
  "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. Esegui la macchina statale.
3. Nella pagina Dettagli di esecuzione dell'esecuzione della macchina a stati, procedi come segue:
   a. Scegliete Causa nel messaggio di errore per visualizzare il motivo dell'errore di esecuzione.
   b. Scegli Mostra i dettagli della fase nel messaggio di errore per visualizzare la fase che ha causato l'errore.
4. Nel messaggio di errore, nota che il contenuto del nodo $.payload.body è una stringa JSON con escape. L'errore si è verificato perché non è possibile fare riferimento a una stringa utilizzando la notazione del percorso JSON.
5. Per fare riferimento al nodo $.payload.body.message, effettuate le seguenti operazioni:
   a. Usa la funzione States.StringToJSON intrinseca per convertire prima la stringa in un formato JSON.
   b. Specifica il percorso JSON per il nodo $.payload.body.message all'interno della funzione intrinseca.

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

6. Esegui nuovamente la macchina a stato.

Casi d'uso

AWS Step Functionsconsente di creare flussi di lavoro visivi che aiutano a tradurre rapidamente i requisiti aziendali in applicazioni. Step Functions gestisce automaticamente lo stato, i checkpoint e i riavvii e fornisce funzionalità integrate per gestire automaticamente errori ed eccezioni. Per comprendere meglio le funzionalità che Step Functions può fornirti, leggi i seguenti casi d'uso:

Argomenti

- Elaborazione dei dati
- Machine learning
- Orchestrazione di microservizi
- Automazione IT e della sicurezza

Elaborazione dei dati

Con l'aumento del volume di dati, provenienti da fonti sempre più diverse, le organizzazioni devono agire rapidamente per elaborare questi dati per garantire decisioni aziendali più rapide e informate. Per elaborare i dati su larga scala, le organizzazioni devono fornire in modo elastico le risorse per gestire le informazioni che ricevono da dispositivi mobili, applicazioni, satelliti, marketing e vendite, archivi di dati operativi, infrastrutture e altro ancora.

Step Functions offre la scalabilità, l'affidabilità e la disponibilità necessarie per gestire con successo i flussi di lavoro di elaborazione dei dati. Puoi gestire milioni di esecuzioni simultanee con Step Functions in quanto è scalabile orizzontalmente e fornisce flussi di lavoro tolleranti agli errori. Elabora i dati più velocemente utilizzando esecuzioni parallele come il tipo di `Parallel` stato di Step Functions o il parallelismo dinamico utilizzando il relativo tipo di stato. `Eseguire la mappatura` Come parte del tuo flusso di lavoro, puoi utilizzare lo `Eseguire la mappatura` stato per eseguire iterazioni su oggetti in un archivio dati statico come un bucket Amazon S3. Step Functions consente inoltre di riprovare facilmente le esecuzioni non riuscite o di scegliere un modo specifico per gestire gli errori senza la necessità di gestire un processo complesso.

A seconda delle esigenze di elaborazione dei dati, Step Functions si integra direttamente con altri servizi di elaborazione dati forniti da, AWS ad esempio AWS Batchper l'elaborazione in batch, Amazon EMR per l'elaborazione di big data, per la preparazione dei dati, Athena AWS Glueper l'analisi dei dati e per l'elaborazione. AWS Lambda

Alcuni esempi dei tipi di flussi di lavoro di elaborazione dei dati che i clienti utilizzano Step Functions includono:

Elaborazione di file, video e immagini

- Prendi una raccolta di file video e convertili in altre dimensioni o risoluzioni ideali per il dispositivo su cui verranno visualizzati, come telefoni cellulari, laptop o televisori.
- Scatta un'ampia raccolta di foto caricate dagli utenti e convertile in miniature o immagini a varie risoluzioni che possono poi essere visualizzate sui siti Web degli utenti.
- Prendi dati semistrutturati, come un file CSV, e combinali con dati non strutturati, come una fattura, per produrre un report aziendale che viene inviato mensilmente agli stakeholder aziendali.
- Prendi i dati di osservazione della Terra raccolti dai satelliti, convertili in formati allineati tra loro e quindi aggiungi altre fonti di dati raccolte sulla terra per ulteriori informazioni.
- Prendi i registri di trasporto dei vari modi di trasporto per i prodotti e cerca le ottimizzazioni utilizzando Monte Carlo Simulations, quindi invia i report alle organizzazioni e alle persone che si affidano a te per spedire le loro merci.

Coordinate i processi di estrazione, trasformazione e caricamento (ETL):

- Combina i record delle opportunità di vendita con i set di dati delle metriche di marketing attraverso una serie di fasi di preparazione dei dati utilizzando AWS Glue e produci report di business intelligence che possono essere utilizzati in tutta l'organizzazione.
- Crea, avvia e termina un cluster Amazon EMR per l'elaborazione di big data.

Elaborazione in batch e carichi di lavoro HPC (High Performance Computing):

- Crea una pipeline di analisi genomica secondaria che elabora le sequenze grezze dell'intero genoma in chiamate di varianti. Allinea i file raw a una sequenza di riferimento e richiama le varianti su un elenco specificato di cromosomi utilizzando il parallelismo dinamico.
- Trova l'efficienza nella produzione del tuo prossimo dispositivo mobile o di altri dispositivi elettronici simulando vari layout utilizzando diversi composti elettrici e chimici. Esegui l'elaborazione in batch di grandi dimensioni dei tuoi carichi di lavoro attraverso varie simulazioni per ottenere il design ottimale.

Machine learning

L'apprendimento automatico consente alle organizzazioni di analizzare rapidamente i dati raccolti per identificare i modelli, quindi prendere decisioni con un intervento umano minimo. L'apprendimento automatico inizia con un set iniziale di dati, noti come dati di addestramento. Questi dati di addestramento aiutano ad aumentare la precisione delle previsioni di un modello di machine learning e fungono da base per l'apprendimento di questo modello. Una volta che il modello è considerato sufficientemente accurato da soddisfare le esigenze aziendali, viene implementato in produzione. Il AWS Step FunctionsData Science Software Development Kit (SDK) è una libreria open source che ti consente di creare facilmente flussi di lavoro che preelaborano i dati, addestrano e quindi pubblicano i tuoi modelli utilizzando Amazon SageMaker e Step Functions.

La preelaborazione dei set di dati esistenti è il modo in cui un'organizzazione spesso crea dati di formazione. Questo metodo aggiunge informazioni, ad esempio etichettando oggetti in un'immagine, annotando testo o elaborando l'audio. Per preelaborare i dati puoi utilizzare AWS Glue o creare un'istanza di SageMaker notebook che esegua l'app Jupyter Notebook. Una volta pronti, i dati possono essere caricati su Amazon S3 per un facile accesso. Man mano che i modelli di machine learning vengono addestrati, puoi apportare modifiche ai parametri di ciascun modello per migliorare la precisione fino a quando non è pronto per l'implementazione.

Step Functions ti consente di orchestrare flussi di lavoro di machine learning end-to-end su. SageMaker Questi flussi di lavoro possono includere la preelaborazione e la post-elaborazione dei dati, l'ingegneria delle funzionalità, la convalida dei dati e la valutazione dei modelli. Una volta che il modello è stato implementato in produzione, è possibile perfezionare e testare nuovi approcci per migliorare continuamente i risultati aziendali. Puoi creare flussi di lavoro pronti per la produzione direttamente in Python oppure puoi utilizzare Step Functions Data Science SDK per copiare quel flusso di lavoro, sperimentare nuove opzioni e inserire il flusso di lavoro raffinato in produzione.

Alcuni tipi di flussi di lavoro di machine learning per i quali i clienti utilizzano Step Functions includono:

Rilevamento frodi

- Identifica e previene il verificarsi di transazioni fraudolente, come le frodi creditizie.
- Rileva e previeni le acquisizioni di account utilizzando modelli di machine learning addestrati.
- Identifica gli abusi promozionali, inclusa la creazione di account falsi, in modo da poter agire rapidamente.

Personalizzazione e consigli

- Consiglia prodotti a clienti mirati in base a ciò che si prevede possa attirare il loro interesse.
- Prevedi se un cliente aggiornerà il proprio account da un piano gratuito a un abbonamento a pagamento.

Arricchimento dei dati

- Utilizza l'arricchimento dei dati come parte della preelaborazione per fornire dati di addestramento migliori per modelli di machine learning più accurati.
- Annota brani di testo e audio per aggiungere informazioni sintattiche, come sarcasmo e gergo.
- Etichetta gli oggetti aggiuntivi nelle immagini per fornire informazioni fondamentali da cui il modello possa imparare, ad esempio se un oggetto è una mela, una palla da basket, una roccia o un animale.

Orchestrazione di microservizi

L'architettura dei microservizi suddivide le applicazioni in servizi vagamente accoppiati. I vantaggi includono una migliore scalabilità, una maggiore resilienza e un time-to-market più rapido. Ogni microservizio è indipendente, il che semplifica la scalabilità di un singolo servizio o funzione senza la necessità di scalare l'intera applicazione. I singoli servizi sono strettamente associati, permettendo ai team indipendenti di concentrarsi su un singolo processo aziendale, senza la necessità di comprendere l'intera applicazione. I microservizi consentono inoltre di scegliere i singoli componenti più adatti alle esigenze aziendali, offrendoti la flessibilità di modificare la selezione senza riscrivere l'intero flusso di lavoro. Team diversi possono utilizzare i linguaggi di programmazione e i framework di loro scelta per lavorare con il proprio microservizio e questo microservizio può comunque comunicare con qualsiasi altro componente dell'applicazione tramite interfacce di programmazione delle applicazioni (API).

Step Functions offre diversi modi per gestire i flussi di lavoro dei microservizi. Per flussi di lavoro a lunga durata, puoi utilizzare i flussi di lavoro standard con l'AWS Fargateintegrazione per orchestrare le applicazioni in esecuzione in contenitori. Per flussi di lavoro di breve durata e ad alto volume che richiedono una risposta immediata, i flussi di lavoro Synchronous Express sono ideali. Questi possono essere utilizzati per applicazioni basate sul Web o mobili, che spesso hanno flussi di lavoro di breve durata e richiedono il completamento di una serie di passaggi prima di restituire una risposta. Puoi attivare un Synchronous Express Workflow direttamente da Amazon API Gateway e la

connessione rimane aperta fino al completamento o al timeout del flusso di lavoro. Per flussi di lavoro di breve durata che non richiedono una risposta immediata, Step Functions fornisce flussi di lavoro rapidi asincroni.

Esempi di alcune orchestrazioni di API che utilizzano Step Functions includono:

Flussi di lavoro sincroni o in tempo reale

- Modifica un valore in un record, ad esempio aggiornando il cognome di un dipendente, e rendi la modifica immediatamente visibile sullo schermo.
- Aggiorna un ordine durante il checkout, ad esempio aggiungendo, rimuovendo o modificando la quantità di un articolo, quindi comunica immediatamente l'aggiornamento al cliente.
- Esegui un processo di elaborazione rapida e restituisci immediatamente il risultato al richiedente.

Orchestrazione dei contenitori

- Esegui processi su Kubernetes con Amazon Elastic Kubernetes Service o su Amazon Elastic Container Service (ECS) con Fargate e integra altri AWS servizi, come l'invio di notifiche con Amazon SNS, come parte dello stesso flusso di lavoro.

Automazione IT e della sicurezza

L'automazione IT può aiutare a gestire operazioni sempre più complesse e dispendiose in termini di tempo, come l'aggiornamento e l'applicazione di patch del software, l'implementazione di aggiornamenti di sicurezza per risolvere le vulnerabilità, la selezione dell'infrastruttura, la sincronizzazione dei dati, il routing dei ticket di supporto e altro ancora. L'automazione di attività ripetitive e dispendiose in termini di tempo può consentire all'organizzazione di completare le operazioni di routine in modo rapido e coerente su larga scala. Ciò consente di concentrarsi su attività strategiche come lo sviluppo di funzionalità, le richieste di supporto complesse e l'innovazione, soddisfacendo al contempo queste crescenti esigenze.

Step Functions ti consente di creare flussi di lavoro che si ridimensionano automaticamente per soddisfare le esigenze della tua azienda senza richiedere interventi manuali. Nei casi in cui si verifica un errore nel flusso di lavoro, spesso non richiede un intervento manuale. Step Functions consente di riprovare automaticamente le attività non riuscite e un backoff esponenziale in grado di gestire gli errori nel flusso di lavoro.

Ci possono essere situazioni in cui è necessario l'intervento umano prima che il flusso di lavoro possa progredire. Ad esempio, l'approvazione di un sostanziale aumento del credito può richiedere l'approvazione umana. Per gestirlo, puoi definire la logica di ramificazione in Step Functions, in modo che solo le richieste superiori a un importo definito richiedano l'approvazione umana, mentre tutte le altre richieste vengano completate automaticamente. Nei casi in cui è richiesta l'approvazione umana, Step Functions consente di sospendere il flusso di lavoro in una fase specifica, attendere una risposta e quindi continuare il flusso di lavoro una volta ricevuta la risposta.

Alcuni esempi dei tipi di flussi di lavoro di automazione per cui i clienti utilizzano Step Functions includono:

Automazione IT

- Risolvi automaticamente incidenti come l'apertura di una porta SSH, lo spazio su disco insufficiente o l'accesso pubblico a un bucket Amazon S3.
- Automatizza l'implementazione di AWS CloudFormation StackSets

Automazione della sicurezza

- Automatizza la risposta a uno scenario in cui un utente e una chiave di accesso utente sono stati esposti.
- Correggi automaticamente le risposte agli incidenti di sicurezza in base alle azioni politiche definite, come la limitazione delle azioni a ARN specifici o l'applicazione di altre azioni.
- Avvisa i dipendenti delle e-mail di phishing entro pochi secondi dalla ricezione.

Approvazione umana

- Automatizza la formazione del modello di machine learning e richiedi l'approvazione manuale del modello da parte di un data scientist prima di implementare o rifiutare automaticamente il modello in base alla risposta ricevuta.
- Automatizza l'invio dei feedback ricevuti dai clienti in base all'analisi del sentiment, in modo che i clienti con un'opinione negativa vengano immediatamente sottoposti a revisione manuale.

Come funziona Step Functions

Questa sezione descrive concetti importanti per aiutarti ad acquisire familiarità con AWS Step Functions e comprendere come funziona.

Argomenti

- Flussi di lavoro Standard e Express
- States
- Modalità di elaborazione dello stato della mappa
- Soglia di errore tollerata per lo stato della mappa distribuita
- Transizioni
- Dati delle macchine a stati
- Elaborazione di input e output in Step Functions
- Simulatore di flusso di dati
- Gestisci le distribuzioni continue con versioni e alias
- Esecuzioni in Step Functions
- Gestione degli errori in Step Functions
- InvocareAWS Step Functionsda altri servizi
- Leggi Coerency in Step Functions
- Inserimento di tag in Step Functions

Flussi di lavoro Standard e Express

Quando si crea una macchina a stati, si seleziona un tipo Standard o Express. Il tipo predefinito per le macchine a stati è Standard. Una macchina a stati il cui tipo è Standard viene denominata flusso di lavoro Standard e una macchina a stati il cui tipo è Express è denominata flusso di lavoro Express.

Sia per i flussi di lavoro Standard che Express, si definisce la macchina a stati utilizzando ilAmazon States Language. Le esecuzioni delle macchine a stati si comporteranno in modo diverso a seconda del tipo selezionato.

> **Important**
>
> Il tipo scelto non può essere modificato dopo aver creato la macchina a stati.

> **Note**
>
> Se si definiscono le macchine a stati al di fuori della console di Step Functions, ad esempio in un editor di propria scelta, è necessario salvare le definizioni delle macchine a stati con l'estensione .asl.json.

I flussi di lavoro standard sono ideali per flussi di lavoro di lunga durata (fino a un anno), durevoli e verificabili. Puoi recuperare la cronologia completa delle esecuzioni utilizzando l'API Step Functions per un massimo di 90 giorni dopo il completamento dell'esecuzione. I flussi di lavoro standard seguono un modello basato sull'esecuzione esatta, in cui le attività e gli stati non vengono mai eseguiti più di una volta, a meno che non sia stato specificato un comportamento in ASL. `Retry` Ciò rende i flussi di lavoro standard adatti a orchestrare azioni non idempotenti, come l'avvio di un cluster Amazon EMR o l'elaborazione dei pagamenti. Le esecuzioni dei flussi di lavoro standard vengono fatturate in base al numero di transizioni di stato elaborate.

I flussi di lavoro Express sono ideali per carichi di lavoro di elaborazione eventi a volume elevato, come l'acquisizione di dati IoT, l'elaborazione e la trasformazione dei dati in streaming e i back-end delle applicazioni mobili. L'esecuzione può durare fino a cinque minuti. I flussi di lavoro Express utilizzano un at-least-oncemodello in cui un'esecuzione potrebbe potenzialmente essere eseguita più di una volta. Ciò rende Express Workflows ideale per orchestrare azioni idempotenti come la trasformazione dei dati di input e l'archiviazione tramite un'azione PUT in Amazon DynamoDB. Le esecuzioni Express Workflow vengono fatturate in base al numero di esecuzioni, alla durata dell'esecuzione e alla memoria consumata durante l'esecuzione.

I flussi di lavoro Standard ed Express possono essere avviati automaticamente in risposta a eventi come richieste HTTP da Amazon API Gateway (API completamente gestite su larga scala), regole IoT e oltre 140 altre fonti di eventi in Amazon. EventBridge

> **Tip**
>
> Per implementare un esempio di flusso di lavoro Express sul tuoAccount AWS, consulta Modulo 7 - API Gateway, Parallel State, Flussi di lavoro Express di The AWS Step Functions Workshop.

Per informazioni sull'esperienza della console per le esecuzioni Standard ed Express Workflow, consulta. Esecuzioni Standard ed Express Workflow nella console

Flussi di lavoro Standard ed Express

	Flussi di lavoro Standard	Flussi di lavoro Express: sincroni e asincroni
Durata massima	Un anno	Cinque minuti
Frequenza di avvio dell'esecuzione supportata	Per informazioni sulle quote relative alla frequenza di avvio dell'esecuzione supportata, vedereQuote relative alla limitazione delle azioni delle API.	Per informazioni sulle quote relative alla frequenza di avvio dell'esecuzione supportata, vedere. Quote relative alla limitazione delle azioni delle API
Frequenza di transizione dello stato supportata	Per informazioni sulle quote relative alla velocità di transizione degli stati supportata, vedere. Quote relative alla limitazione statale	Nessun limite
Prezzi	Prezzo in base al numero di transizioni di stato. Una transizione di stato viene conteggiata ogni volta che viene completata una fase dell'esecuzione.	Il prezzo è calcolato in base al numero di esecuzioni eseguite, alla loro durata e al consumo di memoria.
Cronologia delle esecuzioni	Le esecuzioni possono essere elencate e descritte con	Cronologia di esecuzione illimitata, ovvero vengono

	Flussi di lavoro Standard	Flussi di lavoro Express: sincroni e asincroni
	le API Step Functions. Le esecuzioni possono essere sottoposte a debug visivi tramite la console. Possono anche essere ispezionate in CloudWatch Logs abilitando la registrazione sulla macchina a stati. Per ulteriori informazioni sul debug delle esecuzioni Standard Workflow nella console, consulta e. Esecuzioni Standard ed Express Workflow nella console Visualizzazione ed esecuzione del debug delle esecuzioni	mantenute tutte le voci della cronologia di esecuzione che è possibile generare in un periodo di 5 minuti. Le esecuzioni possono essere controllate in CloudWatch Logs o nella console Step Functions abilitando la registrazione sulla macchina a stati. Per ulteriori informazioni sul debug delle esecuzioni di Express Workflow nella console, consulta e. Esecuzioni Standard ed Express Workflow nella console Visualizzazione ed esecuzione del debug delle esecuzioni
Semantica di esecuzione	Esecuzione del flusso di lavoro esattamente una volta.	Flussi di lavoro Express asincroni: esecuzione di un flusso di lavoro. t-least-once Flussi di lavoro Synchrono us Express: esecuzione di un flusso di lavoro. t-most-once

	Flussi di lavoro Standard	Flussi di lavoro Express: sincroni e asincroni
Integrazioni dei servizi	Supporta tutte le integrazioni e i modelli di servizi.	Supporta tutte le integrazioni di servizi. **Note** I flussi di lavoro Express non supportano i modelli di integrazione dei servizi Job-run (`.sync`) o Callback (`.waitForTaskToken`).
Attività Step Functions	Supporta le attività di Step Functions.	Non supporta le attività di Step Functions.

Flussi di lavoro Express sincroni e asincroni

È possibile scegliere tra due tipi di flussi di lavoro Express: flussi di lavoro Express asincroni e flussi di lavoro Express sincroni.

- I flussi di lavoro Asynchronous Express confermano che il flusso di lavoro è stato avviato, ma non aspettate il completamento del flusso di lavoro. Per ottenere il risultato, è necessario eseguire il polling dei log del servizio. CloudWatch Puoi utilizzare Asynchronous Express Workflows quando non hai bisogno di un output di risposta immediato, come i servizi di messaggistica o l'elaborazione dei dati da cui altri servizi non dipendono. È possibile avviare Asynchronous Express Workflows in risposta a un evento, tramite un flusso di lavoro annidato in Step Functions o utilizzando la chiamata API. `StartExecution`
- Synchronous Express Workflows avvia un flusso di lavoro, attende il completamento e quindi restituisce il risultato. I flussi di lavoro Synchronous Express possono essere utilizzati per orchestrare i microservizi. Con Synchronous Express Workflows, puoi sviluppare applicazioni senza la necessità di sviluppare codice aggiuntivo per gestire errori, tentativi o eseguire attività

parallele. Puoi eseguire Synchronous Express Workflows richiamati da Amazon API Gateway o utilizzando la `StartSyncExecution` chiamata API. AWS Lambda

> **Note**
>
> Se esegui Step Functions Express Workflows in modo sincrono dalla console, la `StartSyncExecution` richiesta scade dopo 60 secondi. Per eseguire Express Workflows in modo sincrono per una durata massima di cinque minuti, effettua la `StartSyncExecution` richiesta utilizzando l'AWSSDK o AWS Command Line Interface (AWS CLI) anziché la console Step Functions.

Le chiamate all'API di esecuzione sincrona di Express non contribuiscono ai limiti di capacità degli account esistenti. Step Functions fornisce capacità su richiesta e si ridimensiona automaticamente con un carico di lavoro sostenuto. I picchi di carico di lavoro possono essere limitati fino a quando la capacità non sarà disponibile.

Garanzie di esecuzione

Flussi di lavoro Standard	Flussi di lavoro Express asincroni	Flussi di lavoro Express sincroni	
Esecuzione del flusso di lavoro esattamente una volta	Un'esecuzione del flusso di lavoro t-least-once	Un'esecuzione t-most-once del flusso di lavoro	
Lo stato di esecuzione persiste internamente tra le transizioni di stato.	Lo stato di esecuzione non persiste tra le transizioni di stato.	Lo stato di esecuzione non persiste tra le transizioni di stato.	
Restituisce automaticamente una risposta idempotente all'avvio di un'esecuzione con lo stesso nome di un workflow attualmen	L'idempotenza non viene gestita automaticamente. L'avvio di più flussi di lavoro con lo stesso nome comporta	L'idempotenza non viene gestita automaticamente. Step Functions attende l'avvio di un'esecuzione e	

Flussi di lavoro Standard	Flussi di lavoro Express asincroni	Flussi di lavoro Express sincroni	
te in esecuzione. Il nuovo flusso di lavoro non si avvia e viene generata un'eccezi one una volta completato il flusso di lavoro attualmente in esecuzione.	esecuzioni simultane e. Può causare la perdita dello stato interno del flusso di lavoro se la logica della macchina a stati non è idempotente.	restituisce il risultato della macchina a stati al termine. I flussi di lavoro non si riavviano se si verifica un'eccezi one.	
Dati della cronologia di esecuzione rimossi dopo 90 giorni. I nomi dei flussi di lavoro possono essere riutilizzati dopo la rimozione dei dati di out-of-date esecuzion e. Per soddisfare i requisiti di conformit à, organizzativi o normativi, è possibile ridurre il periodo di conservazione della cronologia di esecuzione a 30 giorni inviando una richiesta di quota. A tale scopo, utilizza AWS Support Center Console e crea un nuovo caso.	La cronologia delle esecuzioni non viene acquisita da Step Functions. La registrazione deve essere abilitata tramite Amazon CloudWatch Logs.	La cronologia delle esecuzioni non viene acquisita da Step Functions. La registrazione deve essere abilitata tramite Amazon CloudWatch Logs.	

Ottimizzazione dei costi con Express Workflows

Step Functions determina i prezzi per i flussi di lavoro Standard ed Express in base al tipo di flusso di lavoro utilizzato per creare le macchine a stati. Per ottimizzare il costo dei flussi di lavoro serverless, puoi seguire uno o entrambi i seguenti consigli:

Argomenti

- Suggerimento #1: annidamento dei flussi di lavoro Express all'interno dei flussi di lavoro Standard
- Suggerimento #2: converti i flussi di lavoro standard in flussi di lavoro Express

Per informazioni su come la scelta di un tipo di flusso di lavoro Standard o Express influisce sulla fatturazione, consulta Prezzi. AWS Step Functions

Suggerimento #1: annidamento dei flussi di lavoro Express all'interno dei flussi di lavoro Standard

Step Functions esegue flussi di lavoro con una durata e un numero di passaggi limitati. Alcuni flussi di lavoro possono completare l'esecuzione entro un breve periodo di tempo. Altri potrebbero richiedere una combinazione di flussi di lavoro e high-event-rate di lunga durata. Con Step Functions, puoi creare flussi di lavoro ampi e complessi partendo da più flussi di lavoro più piccoli e semplici.

Ad esempio, per creare un flusso di lavoro di elaborazione degli ordini, puoi includere tutte le azioni non idempotenti in un flusso di lavoro Standard. Ciò potrebbe includere azioni, come l'approvazione dell'ordine tramite l'interazione umana e l'elaborazione dei pagamenti. È quindi possibile combinare una serie di azioni idempotenti, come l'invio di notifiche di pagamento e l'aggiornamento dell'inventario dei prodotti, in un flusso di lavoro Express. È possibile inserire questo flusso di lavoro Express all'interno del flusso di lavoro Standard. In questo esempio, il flusso di lavoro Standard è noto come macchina a stati principale. Il flusso di lavoro Express annidato è noto come macchina a stati secondari.

Suggerimento #2: converti i flussi di lavoro standard in flussi di lavoro Express

È possibile convertire i flussi di lavoro Standard esistenti in flussi di lavoro Express se soddisfano i seguenti requisiti:

- L'esecuzione del flusso di lavoro deve essere completata entro cinque minuti.
- Il flusso di lavoro è conforme a un modello di at-least-onceesecuzione. Ciò significa che ogni fase del flusso di lavoro può essere eseguita più di una volta esatta.

- Il flusso di lavoro non utilizza i modelli di integrazione dei `.sync` servizi `.waitForTaskToken` o dei servizi.

> **Important**
>
> I flussi di lavoro Express utilizzano Amazon CloudWatch Logs per registrare le cronologie di esecuzione. Dovrai sostenere costi aggiuntivi quando utilizzi Logs. CloudWatch

Per convertire un flusso di lavoro Standard in un flusso di lavoro Express utilizzando la console

1. Apri la console Step Functions.
2. Nella pagina Macchine a stati, scegli una macchina a stati di tipo Standard per aprirla.

   > **Tip**
   >
   > Dall'elenco a discesa Qualsiasi tipo, scegli Standard per filtrare l'elenco delle macchine a stati e visualizzare solo i flussi di lavoro Standard.

3. Scegli Copia su nuovo.

   Workflow Studio si apre con la Modalità di progettazione visualizzazione del flusso di lavoro della macchina a stati selezionata.
4. (Facoltativo) Aggiornate il design del flusso di lavoro.
5. Specificate un nome per la vostra macchina a stati. Per fare ciò, scegli l'icona di modifica accanto al nome della macchina a stati predefinita di MyStateMachine. Quindi, nella configurazione della macchina a stati, specifica un nome nella casella Nome macchina a stati.
6. (Facoltativo) Nella configurazione della macchina a stati, specificate altre impostazioni del flusso di lavoro, come il tipo di macchina a stati e il relativo ruolo di esecuzione.

   Assicurati che per Type, scegli Express. Mantieni tutte le altre selezioni predefinite nelle impostazioni della macchina a stati.

   > **Note**
   >
   > Se state convertendo un flusso di lavoro standard precedentemente definito in AWS CDKorAWS SAM, dovete modificare il valore `Type` e `Resource` il nome.

7. Nella finestra di dialogo Conferma creazione del ruolo, scegli Conferma per continuare.

   Puoi anche scegliere Visualizza le impostazioni del ruolo per tornare alla configurazione della macchina a stati.

   > **Note**
   >
   > Se elimini il ruolo IAM creato da Step Functions, Step Functions non può ricrearlo in un secondo momento. Allo stesso modo, se modifichi il ruolo (ad esempio, rimuovendo Step Functions dai principi nella policy IAM), Step Functions non può ripristinare le impostazioni originali in un secondo momento.

Per ulteriori informazioni sulle migliori pratiche e linee guida per la gestione dell'ottimizzazione dei costi per i flussi di lavoro, consulta Creazione di flussi di lavoro convenienti. AWS Step Functions

States

I singoli stati possono prendere decisioni in base ai propri input, eseguire azioni da tali input e trasmettere l'output ad altri stati. InAWS Step Functions, definisci i tuoi flussi di lavoro in Amazon States Language (ASL). La console Step Functions fornisce una rappresentazione grafica della macchina a stati per facilitare la visualizzazione della logica dell'applicazione.

> **Note**
>
> Se definisci le tue macchine a stato all'esterno della console di Step Functions, ad esempio in un editor di tua scelta, devi salvare le definizioni delle tue macchine a stato con l'estensione .asl.json.

Gli stati sono elementi della tua macchina statale. Per fare riferimento a uno stato ne viene utilizzato il nome, che può essere costituito da qualsiasi stringa, ma deve essere univoco nell'ambito dell'intera macchina a stati.

Nella macchina a stati, gli stati possono svolgere diverse funzioni.

- Eseguire operazioni nella macchina a stati (stato Task)
- Effettuare una scelta tra rami di esecuzione (stato Choice)

- Arrestare un'esecuzione con esito negativo o positivo (stato Fail o Succeed)
- Passa l'input all'output o inserisci alcuni dati fissi nel flusso di lavoro (uno stato Pass)
- Fornisci un ritardo per un determinato periodo di tempo o fino a una data e ora specificate (uno stato di attesa)
- Avviare rami di esecuzione paralleli (stato Parallel)
- Iterare dinamicamente le fasi (stato Map)

Di seguito è riportato uno stato di esempio denominato `HelloWorld` che esegue una funzione AWS Lambda.

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

Gli stati condividono svariate caratteristiche comuni:

- Un `Type` campo che indica il tipo di stato.
- Un `Comment` campo opzionale per contenere un commento leggibile dall'uomo sullo stato o una descrizione dello stato.
- Ogni stato (tranne gli stati `Succeed` e `Fail`) deve includere un campo `Next` oppure, in alternativa, specificare un campo `End` in modo da diventare uno stato terminale.

> **Note**
>
> Uno stato `Choice` può includere più campi `Next`, ma solo uno all'interno di ciascuna regola Choice. Uno `Choice` stato non può usare`End`.

Determinati tipi di stato richiedono campi aggiuntivi o potrebbero ridefinire l'utilizzo di un campo comune.

Dopo aver creato ed eseguito i flussi di lavoro Standard, puoi accedere alle informazioni su ogni stato, i relativi input e output, quando è stato attivo e per quanto tempo, visualizzando la

pagina Dettagli dell'esecuzione nella console Step Functions. Per ulteriori informazioni, consulta Visualizzazione e debug delle esecuzioni sulla console Step Functions.

Dopo aver creato ed eseguito le esecuzioni di Express Workflow e se la registrazione è abilitata per Express Workflow, puoi accedere alle informazioni sull'esecuzione in Amazon CloudWatch Logs o nella console Step Functions. Per ulteriori informazioni, consulta Visualizzazione e debug delle esecuzioni sulla console Step Functions.

Argomenti

- Amazon States Language
- Pass
- Attività
- Choice
- Attendi
- Succeed
- Fail
- Parallel
- Eseguire la mappatura

Amazon States Language

Amazon States Language è un linguaggio strutturato basato su JSON utilizzato per definire la tua macchina a stati, un insieme di stati che può funzionare (`Task`stati), determinare quali stati passare a quelli successivi (`Choice`stati), interrompere un'esecuzione con un errore (`Fail`stati) e così via.

Per ulteriori informazioni, consulta la Amazon States Language Specification (Specifica del linguaggio degli stati di Amazon) e Statelint, uno strumento che consente di convalidare il codice di Amazon States Language.

Per creare una macchina a stati sulla console Step Functions utilizzando Amazon States Language, consulta Guida introduttiva.

> **Note**
>
> Se definisci le tue macchine a stato all'esterno della console di Step Functions, ad esempio in un editor di tua scelta, devi salvare le definizioni delle tue macchine a stato con l'estensione .asl.json.

Esempio di specifiche linguistiche degli Stati Amazon

```
{
  "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
    }
  }
}
```

Argomenti

- Struttura della macchina a stati
- Funzioni intrinseche
- Campi di stato comuni

Struttura della macchina a stati

Le macchine a stati vengono definite utilizzando testo JSON che rappresenta una struttura contenente i seguenti campi.

`Comment` (facoltativo).

Una descrizione leggibile della macchina a stati.

`StartAt` (Obbligatorio)

Una stringa che deve corrispondere esattamente (con distinzione tra maiuscole e minuscole) al nome di uno degli oggetti di stato.

`TimeoutSeconds`(Facoltativo)

Il numero massimo di secondi durante i quali può essere eseguita la macchina a stati. Se viene eseguita per un tempo superiore a quello specificato, l'esecuzione ha esito negativo con un `States.Timeout` nome di errore.

`Version` (facoltativo).

La versione di Amazon States Language utilizzata nella macchina a stati (l'impostazione predefinita è «1.0").

`States` (Obbligatorio)

Un oggetto contenente un set di stati delimitati da virgola.

Il campo `States` contiene stati.

```
{
    "State1" : {
    },

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

Una macchina a stati è definita dagli stati che contiene e dalle relazioni tra gli stessi.

Di seguito è riportato un esempio.

```
{
  "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
    }
  }
}
```

Quando viene avviata un'esecuzione di questa macchina a stati, il sistema inizia con lo stato cui viene fatto riferimento nel campo `StartAt` (`"HelloWorld"`). Se questo stato ha un campo `"End": true`, l'esecuzione viene arrestata e restituisce un risultato. In caso contrario, il sistema cerca un campo `"Next":` e continua con quello stato. Questo processo si ripete finché il sistema non raggiunge uno stato terminale, ovvero uno stato con `"Type": "Succeed"`, `"Type": "Fail"` o `"End": true`, oppure finché non si verifica un errore di runtime.

Le regole seguenti si applicano agli stati in una macchina a stati:

- Gli stati possono verificarsi in qualsiasi ordine nel blocco di inclusione, ma l'ordine in cui sono elencati non influenza l'ordine in cui vengono eseguiti, che dipende dal contenuto degli stati stessi.
- Una macchina a stati può includere un solo stato designato come stato `start` tramite il valore del campo `StartAt` nella struttura di livello superiore. Questo stato è quello che viene eseguito per primo quando viene avviata l'esecuzione.
- Qualsiasi stato per il quale il campo `End` è `true` viene considerato uno stato `end` (o `terminal`). A seconda della logica della macchina a stati, ad esempio se la macchina a stati ha più rami di esecuzione, potresti avere più di uno stato. `end`
- Se la macchina a stati include un solo stato, può essere contemporaneamente lo stato `start` e lo stato `end`.

Funzioni intrinseche

Amazon States Language offre diverse funzioni intrinseche, note anche come intrinseche, che ti aiutano a eseguire operazioni di elaborazione dei dati di base senza utilizzare uno stato. `Task` Gli intrinseci sono costrutti simili alle funzioni dei linguaggi di programmazione. Possono essere usati per aiutare i costruttori di payload a elaborare i dati da e verso il campo di uno stato. `Resource` `Task`

In Amazon States Language, le funzioni intrinseche sono raggruppate nelle seguenti categorie, in base al tipo di attività di elaborazione dei dati che desideri eseguire:

- Elementi intrinseci per gli array
- Intrinseci per la codifica e la decodifica dei dati
- Intrinseco per il calcolo dell'hash
- Caratteristiche intrinseche per la manipolazione dei dati JSON
- Elementi intrinseci per le operazioni matematiche
- Intrinseco per il funzionamento con stringhe

- Intrinseco per la generazione di identificatori univoci
- Intrinseco per operazioni generiche

> **Note**
>
> - Per utilizzare le funzioni intrinseche devi specificare il valore chiave `.$` nelle definizioni della tua macchina a stati, come mostrato nell'esempio seguente:
>
> ```
> "KeyId.$": "States.Array($.Id)"
> ```
>
> - È possibile inserire fino a 10 funzioni intrinseche all'interno di un campo nei flussi di lavoro. L'esempio seguente mostra un campo denominato *myArn* che include nove funzioni intrinseche annidate:
>
> ```
> "myArn.$": "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**
>
> Se utilizzi Step Functions in un ambiente di sviluppo locale, assicurati di utilizzare la versione 1.12.0 o successiva per poter includere tutte le funzioni intrinseche nei flussi di lavoro.

Campi che supportano le funzioni intrinseche

La tabella seguente mostra quali campi supportano le funzioni intrinseche per ogni stato.

Campi che supportano le funzioni intrinseche

State	Passa	Attività	Scelta	Attendi	Avere successo	Fallire	Parallela	Mappa
InputPath								
Parametri	✓	✓					✓	✓
ResultSelector		✓					✓	✓
ResultPath								
OutputPath								
Variabile								
<ComparisonOperator> Percorso								
TimeoutSecondsPath								
HeartbeatSecondsPath								
Credenziali		✓						

Elementi intrinseci per gli array

Utilizzate i seguenti elementi intrinseci per eseguire manipolazioni degli array.

`States.Array`

La funzione `States.Array` intrinseca accetta zero o più argomenti. L'interprete restituisce un array JSON contenente i valori degli argomenti nell'ordine fornito. Ad esempio, con il seguente input:

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

Potresti usare

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

Ciò restituirebbe il seguente risultato:

```
“BuildId”: [123456]
```

`States.ArrayPartition`

Usa la funzione `States.ArrayPartition` intrinseca per partizionare un array di grandi dimensioni. È inoltre possibile utilizzare questa funzionalità intrinseca per suddividere i dati e quindi inviare il payload in blocchi più piccoli.

Questa funzione intrinseca richiede due argomenti. Il primo argomento è un array, mentre il secondo argomento definisce la dimensione del blocco. L'interprete suddivide l'array di input in più array della dimensione specificata dalla dimensione del blocco. La lunghezza dell'ultimo blocco dell'array può essere inferiore alla lunghezza dei blocchi dell'array precedente se il numero di elementi rimanenti nell'array è inferiore alla dimensione del blocco.

Convalida dell'input

- È necessario specificare un array come valore di input per il primo argomento della funzione.
- È necessario specificare un numero intero positivo diverso da zero per il secondo argomento che rappresenta il valore della dimensione del blocco.

  Se specificate un valore non intero per il secondo argomento, Step Functions lo arrotonderà al numero intero più vicino.

- L'array di input non può superare il limite di dimensione del payload di Step Functions di 256 KB.

Ad esempio, dato il seguente array di input:

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

È possibile utilizzare la `States.ArrayPartition` funzione per dividere l'array in blocchi di quattro valori:

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

Ciò restituirebbe i seguenti blocchi di array:

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

Nell'esempio precedente, la `States.ArrayPartition` funzione emette tre array. I primi due array contengono ciascuno quattro valori, come definito dalla dimensione del blocco. Un terzo array contiene il valore rimanente ed è inferiore alla dimensione del blocco definita.

`States.ArrayContains`

Utilizzate la funzione `States.ArrayContains` intrinseca per determinare se un valore specifico è presente in un array. Ad esempio, è possibile utilizzare questa funzione per rilevare se si è verificato un errore in un'iterazione di `Map` stato.

Questa funzione intrinseca richiede due argomenti. Il primo argomento è un array, mentre il secondo è il valore da cercare all'interno dell'array.

Convalida dell'input

- È necessario specificare un array come valore di input per il primo argomento della funzione.
- È necessario specificare un oggetto JSON valido come secondo argomento.
- L'array di input non può superare il limite di dimensione del payload di Step Functions di 256 KB.

Ad esempio, dato il seguente array di input:

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

```
}
```

È possibile utilizzare la `States.ArrayContains` funzione per trovare il `lookingFor` valore all'interno di`inputArray`:

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

Poiché il valore memorizzato in `lookingFor` è incluso in`inputArray`, `States.ArrayContains` restituisce il seguente risultato:

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

`States.ArrayRange`

Utilizzate la funzione `States.ArrayRange` intrinseca per creare un nuovo array contenente un intervallo specifico di elementi. Il nuovo array può contenere fino a 1000 elementi.

Questa funzione accetta tre argomenti. Il primo argomento è il primo elemento del nuovo array, il secondo argomento è l'elemento finale del nuovo array e il terzo argomento è il valore di incremento tra gli elementi del nuovo array.

Convalida dell'input

- È necessario specificare valori interi per tutti gli argomenti.

  Se specificate un valore non intero per uno qualsiasi degli argomenti, Step Functions lo arrotonderà al numero intero più vicino.
- È necessario specificare un valore diverso da zero per il terzo argomento.
- L'array appena generato non può contenere più di 1000 elementi.

Ad esempio, il seguente utilizzo della `States.ArrayRange` funzione creerà una matrice con un primo valore di 1, un valore finale di 9 e i valori compresi tra il primo e il valore finale aumenteranno di due per ogni elemento:

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

Ciò restituirebbe il seguente array:

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

`States.ArrayGetItem`

Questa funzione intrinseca restituisce il valore di un indice specificato. Questa funzione accetta due argomenti. Il primo argomento è una matrice di valori e il secondo argomento è l'indice dell'array del valore da restituire.

Ad esempio, utilizzate `inputArray` i seguenti `index` valori:

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

Da questi valori, è possibile utilizzare la `States.ArrayGetItem` funzione per restituire il valore nella `index` posizione 5 all'interno dell'array:

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

In questo esempio, `States.ArrayGetItem` restituirebbe il seguente risultato:

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

`States.ArrayLength`

La funzione `States.ArrayLength` intrinseca restituisce la lunghezza di un array. Ha un argomento, l'array di cui restituire la lunghezza.

Ad esempio, dato il seguente array di input:

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

È possibile utilizzare `States.ArrayLength` per restituire la lunghezza di`inputArray`:

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

In questo esempio, `States.ArrayLength` restituirebbe il seguente oggetto JSON che rappresenta la lunghezza dell'array:

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

`States.ArrayUnique`

La funzione `States.ArrayUnique` intrinseca rimuove i valori duplicati da una matrice e restituisce una matrice contenente solo elementi unici. Questa funzione accetta come unico argomento un array, che può essere disordinato.

Ad esempio, quanto segue `inputArray` contiene una serie di valori duplicati:

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

È possibile utilizzare la `States.ArrayUnique` funzione come e specificare l'array da cui rimuovere i valori duplicati:

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

La `States.ArrayUnique` funzione restituirebbe il seguente array contenente solo elementi unici, rimuovendo tutti i valori duplicati:

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

Intrinseci per la codifica e la decodifica dei dati

Utilizzate le seguenti funzioni intrinseche per codificare o decodificare i dati in base allo schema di codifica Base64.

`States.Base64Encode`

Utilizzate la funzione `States.Base64Encode` intrinseca per codificare i dati in base allo schema di codifica MIME Base64. È possibile utilizzare questa funzione per passare dati ad altri servizi senza utilizzare alcuna funzione. AWS AWS Lambda

Questa funzione richiede la codifica di una stringa di dati di massimo 10.000 caratteri come unico argomento.

Ad esempio, si consideri la seguente `input` stringa:

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

È possibile utilizzare la `States.Base64Encode` funzione per codificare la `input` stringa come stringa MIME Base64:

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

La `States.Base64Encode` funzione restituisce i seguenti dati codificati in risposta:

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

`States.Base64Decode`

Usa la funzione `States.Base64Decode` intrinseca per decodificare i dati in base allo schema di decodifica MIME Base64. È possibile utilizzare questa funzione per passare dati ad altri AWS servizi senza utilizzare una funzione Lambda.

Questa funzione accetta come unico argomento una stringa di dati codificati in Base64 contenente fino a 10.000 caratteri da decodificare.

Ad esempio, con il seguente input:

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

È possibile utilizzare la `States.Base64Decode` funzione per decodificare la stringa base64 in una stringa leggibile dall'uomo:

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

In risposta `States.Base64Decode function` restituirebbe i seguenti dati decodificati:

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

Intrinseco per il calcolo dell'hash

`States.Hash`

Usa la funzione `States.Hash` intrinseca per calcolare il valore hash di un dato input. È possibile utilizzare questa funzione per passare dati ad altri AWS servizi senza utilizzare una funzione Lambda.

Questa funzione accetta due argomenti. Il primo argomento riguarda i dati di cui si desidera calcolare il valore hash. Il secondo argomento è l'algoritmo di hashing da utilizzare per eseguire il calcolo dell'hash. I dati forniti devono essere una stringa di oggetto contenente al massimo 10.000 caratteri.

L'algoritmo di hashing specificato può essere uno dei seguenti algoritmi:

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

Ad esempio, è possibile utilizzare questa funzione per calcolare il valore hash della `Data` stringa utilizzando quanto specificato: `Algorithm`

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

È possibile utilizzare la `States.Hash` funzione per calcolare il valore hash:

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

La `States.Hash` funzione restituisce il seguente valore hash in risposta:

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

Caratteristiche intrinseche per la manipolazione dei dati JSON

Utilizzate queste funzioni per eseguire operazioni di elaborazione dei dati di base sugli oggetti JSON.

`States.JsonMerge`

Utilizzate la funzione `States.JsonMerge` intrinseca per unire due oggetti JSON in un unico oggetto. Questa funzione accetta tre argomenti. I primi due argomenti sono gli oggetti JSON che desiderate unire. Il terzo argomento è un valore booleano di. `false` Questo valore booleano determina se la modalità di fusione profonda è abilitata.

Attualmente, Step Functions supporta solo la modalità di fusione superficiale; pertanto, è necessario specificare il valore booleano come. `false` In modalità superficiale, se la stessa chiave esiste in entrambi gli oggetti JSON, la chiave di quest'ultimo oggetto sovrascrive la stessa chiave nel primo oggetto. Inoltre, gli oggetti annidati all'interno di un oggetto JSON non vengono uniti quando si utilizza la fusione superficiale.

Ad esempio, è possibile utilizzare la `States.JsonMerge` funzione per unire i seguenti oggetti JSON che condividono la chiave. a

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

È possibile specificare gli oggetti json1 e json2 come input nella funzione per unirli insieme: `States.JsonMerge`

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

`States.JsonMerge`restituisce come risultato il seguente oggetto JSON unito. Nell'oggetto JSON unito`output`, la chiave dell'oggetto a sostituisce la chiave dell'`json2`oggetto. `json1` a Inoltre, l'oggetto nidificato nella chiave dell'`json1`oggetto a viene scartato perché la modalità superficiale non supporta l'unione di oggetti nidificati.

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

`States.StringToJson`

La `States.StringToJson` funzione utilizza come unico argomento un percorso di riferimento a una stringa JSON con escape.

L'interprete applica un parser JSON e restituisce il formato JSON analizzato dell'input. Ad esempio, è possibile utilizzare questa funzione per sfuggire alla seguente stringa di input:

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

Utilizzate la `States.StringToJson` funzione e specificate `escapedJsonString` come argomento di input:

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

La `States.StringToJson` funzione restituisce il seguente risultato:

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

`States.JsonToString`

La `States.JsonToString` funzione accetta un solo argomento, ovvero il Path che contiene i dati JSON da restituire come stringa senza escape. L'interprete restituisce una stringa che contiene testo JSON che rappresenta i dati specificati dal Path. Ad esempio, puoi fornire il seguente percorso JSON contenente un valore di escape:

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

Fornisci alla `States.JsonToString` funzione i dati contenuti all'interno di: `unescapedJson`

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

La `States.JsonToString` funzione restituisce la seguente risposta:

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

Elementi intrinseci per le operazioni matematiche

Utilizzate queste funzioni per eseguire operazioni matematiche.

`States.MathRandom`

Utilizzate la funzione `States.MathRandom` intrinseca per restituire un numero casuale compreso tra il numero iniziale (incluso) e il numero finale specificato (esclusivo).

È possibile utilizzare questa funzione per distribuire un'attività specifica tra due o più risorse.

Questa funzione richiede tre argomenti. Il primo argomento è il numero iniziale, il secondo argomento è il numero finale e l'ultimo argomento controlla il valore iniziale. L'argomento del valore iniziale è facoltativo. Se si utilizza questa funzione con lo stesso valore iniziale, restituisce un numero identico.

> **Important**
>
> Poiché la `States.MathRandom` funzione non restituisce numeri casuali crittograficamente sicuri, si consiglia di non utilizzarla per applicazioni sensibili alla sicurezza.

Convalida dell'input

- È necessario specificare valori interi per gli argomenti del numero iniziale e del numero finale.

  Se specificate un valore non intero per l'argomento del numero iniziale o del numero finale, Step Functions lo arrotonderà al numero intero più vicino.

Ad esempio, per generare un numero casuale compreso tra uno e 999, è possibile utilizzare i seguenti valori di input:

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

Per generare il numero casuale, fornite i end valori `start` e alla `States.MathRandom` funzione:

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

La `States.MathRandom` funzione restituisce il seguente numero casuale come risposta:

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

`States.MathAdd`

Utilizzate la funzione `States.MathAdd` intrinseca per restituire la somma di due numeri. Ad esempio, è possibile utilizzare questa funzione per incrementare i valori all'interno di un ciclo senza richiamare una funzione Lambda.

Convalida dell'input

- È necessario specificare valori interi per tutti gli argomenti.

  Se specificate un valore non intero per uno o entrambi gli argomenti, Step Functions lo arrotonderà al numero intero più vicino.
- È necessario specificare valori interi nell'intervallo di -2147483648 e 2147483647.

Ad esempio, è possibile utilizzare i seguenti valori per sottrarre uno da 111:

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

Quindi, utilizzate la `States.MathAdd` funzione di definizione `value1` come valore iniziale e `step` come valore da `value1` incrementare di:

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

La `States.MathAdd` funzione restituirebbe il seguente numero in risposta:

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

Intrinseco per il funzionamento con stringhe

`States.StringSplit`

Usa la funzione `States.StringSplit` intrinseca per dividere una stringa in una matrice di valori. Questa funzione accetta due argomenti. Il primo argomento è una stringa e il secondo argomento è il carattere delimitante che la funzione utilizzerà per dividere la stringa.

Example - Divide una stringa di input utilizzando un singolo carattere di delimitazione

Per questo esempio, usa `States.StringSplit` per dividere quanto segue`inputString`, che contiene una serie di valori separati da virgole:

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

Utilizzate la `States.StringSplit` funzione e definite `inputString` come primo argomento e il carattere delimitante `splitter` come secondo argomento:

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

La `States.StringSplit` funzione restituisce come risultato il seguente array di stringhe:

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

Example - Divide una stringa di input utilizzando più caratteri di delimitazione

Per questo esempio, usa `States.StringSplit` per dividere quanto segue`inputString`, che contiene più caratteri di delimitazione:

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

Utilizzate la `States.StringSplit` funzione come segue:

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

La `States.StringSplit` funzione restituisce come risultato il seguente array di stringhe:

```
{"myStringArray": [
  "This",
  "is",
```

```
  "a",
  "test",
  "string"
]}
```

Intrinseco per la generazione di identificatori univoci

`States.UUID`

Utilizza la funzione `States.UUID` intrinseca per restituire un identificatore univoco universale versione 4 (UUID v4) generato utilizzando numeri casuali. Ad esempio, è possibile utilizzare questa funzione per chiamare altri AWS servizi o risorse che richiedono un parametro UUID o inserire elementi in una tabella DynamoDB.

La `States.UUID` funzione viene chiamata senza argomenti specificati:

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

La funzione restituisce un UUID generato casualmente, come nell'esempio seguente:

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

Intrinseco per operazioni generiche

`States.Format`

Usa la funzione `States.Format` intrinseca per costruire una stringa da valori letterali e interpolati. Questa funzione accetta uno o più argomenti. Il valore del primo argomento deve essere una stringa e può includere zero o più istanze della sequenza `{}` di caratteri. Nell'invocazione dell'intrinseco devono esserci tanti argomenti rimanenti quante sono le occorrenze di. `{}` L'interprete restituisce la stringa definita nel primo argomento, ciascuna `{}` sostituita dal valore dell'argomento corrispondente alla posizione nell'invocazione Intrinsic.

Ad esempio, è possibile utilizzare i seguenti input di una persona e una frase in cui inserire il nomename: `template`

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

```
}
```

Utilizzate la `States.Format` funzione e specificate la `template` stringa e la stringa da inserire al posto dei `{}` caratteri:

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

oppure

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

Con uno degli input precedenti, la `States.Format` funzione restituisce la stringa completata in risposta:

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

Caratteri riservati nelle funzioni intrinseche

I seguenti caratteri sono riservati alle funzioni intrinseche e devono essere eliminati con una barra rovesciata ('\') se si desidera che compaiano nel Value:, e. ' { } \

Se il carattere \ deve apparire come parte del valore senza fungere da carattere di escape, devi evitarlo con una barra rovesciata. Le seguenti sequenze di caratteri con escape vengono utilizzate con funzioni intrinseche:

- La stringa letterale rappresenta. `\' '`
- La stringa `\{` letterale rappresenta. `{`
- La stringa `\}` letterale rappresenta. `}`
- La stringa `\\` letterale rappresenta. `\`

In JSON, le barre rovesciate contenute in un valore letterale di stringa devono essere sostituite con un'altra barra rovesciata. L'elenco equivalente per JSON è:

- La stringa `\\\'` di escape rappresenta. `\'`
- La stringa `\\\{` di escape rappresenta. `\{`
- La stringa `\\\}` di escape rappresenta. `\}`

- La stringa \\\\ di escape rappresenta. \\

> **Note**
>
> Se nella stringa di invocazione intrinseca \ viene trovata una barra rovesciata di escape aperta, l'interprete restituirà un errore di runtime.

Campi di stato comuni

`Type` (Obbligatorio)

Il tipo di stato.

`Next`

Il nome dello stato successivo che viene eseguito al termine dello stato corrente. Alcuni tipi di stato, ad esempio `Choice`, consentono più stati di transizione.

Se lo stato corrente è l'ultimo stato del flusso di lavoro o uno stato terminale, ad esempio Succeed oFail, non è necessario specificare il `Next` campo.

`End`

Designa questo stato come stato terminale (termina l'esecuzione) se impostato su `true`. È possibile avere un numero qualsiasi di stati terminali per macchina a stati. Uno solo tra `Next` o `End` può essere utilizzato in uno stato. Alcuni tipi di stato, ad esempio`Choice`, o stati terminali, come Succeed eFail, non supportano né utilizzano il `End` campo.

`Comment` (facoltativo).

Contiene una descrizione leggibile dello stato.

`InputPath` (facoltativo).

Un percorso che seleziona una parte dell'input dello stato da passare al task dello stato per l'elaborazione. Se non specificato, ha il valore predefinito $, che designa l'intero input. Per ulteriori informazioni, consultare Elaborazione di input e output.

`OutputPath` (facoltativo).

Un percorso che seleziona una parte dell'output dello stato da passare allo stato successivo. Se omesso, ha il valore $ che indica l'intero output. Per ulteriori informazioni, consultare Elaborazione di input e output.

Pass

Uno stato `Pass` (`"Type": "Pass"`) passa il relativo input all'output, senza eseguire alcun lavoro. Gli stati `Pass` sono utili durante la costruzione e il debug di macchine a stati.

Puoi anche utilizzare uno `Pass` stato per trasformare l'input di stato JSON utilizzando filtri e quindi passare i dati trasformati allo stato successivo nei tuoi flussi di lavoro. Per informazioni sulla trasformazione dell'input, vedereInputPath, Parametri e ResultSelector.

Oltre ai campi di stato comuni, gli stati `Pass` consentono i seguenti campi.

`Result` (facoltativo).

Si riferisce all'output di un'attività virtuale che viene passata allo stato successivo. Se si include il `ResultPath` campo nella definizione della macchina a stati, `Result` viene inserito come specificato `ResultPath` e passato allo stato successivo.

`ResultPath` (facoltativo).

Specifica dove posizionare l'output (relativo all'input) dell'attività virtuale specificata in`Result`. L'input viene filtrato ulteriormente come specificato dal campo `OutputPath` (se presente) prima di essere utilizzato come output dello stato. Per ulteriori informazioni, consultare Elaborazione di input e output.

`Parameters` (facoltativo).

Crea una raccolta di coppie chiave-valore che verranno passate come input. È `Parameters` possibile specificare un valore statico o selezionarlo dall'input utilizzando un percorso. Per ulteriori informazioni, consulta InputPath, Parametri e ResultSelector.

Esempio di stato Pass

Di seguito viene fornito un esempio di stato `Pass` che introduce alcuni dati fissi nella macchina a stati, probabilmente per scopi di test.

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

```
  "End": true
}
```

Supponiamo che l'input a questo stato sia il seguente.

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

L'output sarebbe quindi questo.

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

Attività

Uno stato `Task` (`"Type": "Task"`) rappresenta una singola unità di lavoro eseguita da una macchina a stati. Un'attività esegue il lavoro utilizzando un'attività o una AWS Lambda funzione, integrandosi con altre supportate o Servizi AWS richiamando un'API di terze parti, come Stripe.

Amazon States Language rappresenta le attività impostando il tipo di stato `Task` e fornendo all'attività l'Amazon Resource Name (ARN) dell'attività, la funzione Lambda o l'endpoint API di terze parti. La seguente definizione dello stato dell'attività richiama una funzione Lambda denominata. *`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
}
```

In questo argomento

- Tipi di attività
- Campi relativi allo stato delle attività
- Esempi di definizione dello stato dell'attività
- Attività

Tipi di attività

Step Functions supporta i seguenti tipi di attività che è possibile specificare in una definizione dello stato dell'attività:

- Attività
- Funzioni Lambda
- A supportato Servizio AWS
- Un'attività HTTP

È possibile specificare un tipo di attività fornendo il relativo ARN nel `Resource` campo di una definizione dello stato dell'attività. L'esempio seguente mostra la sintassi del `Resource` campo. Tutti i tipi di Task, tranne quello che richiama un'API di terze parti, utilizzano la seguente sintassi. Per informazioni sulla sintassi del task HTTP, vedere. Chiama API di terze parti

Nella definizione dello stato del task, sostituisci il testo in corsivo nella sintassi seguente con le informazioni specifiche della risorsa. AWS

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

L'elenco seguente illustra i singoli componenti di questa sintassi:

- `partition` è la partizione AWS Step Functions da utilizzare, più comunemente `aws`.
- `service`indica il valore Servizio AWS utilizzato per eseguire l'operazione e può corrispondere a uno dei seguenti valori:
  - `states` per un'attività
  - `lambda`per una funzione Lambda. Se esegui l'integrazione con altriServizi AWS, ad esempio Amazon SNS o Amazon DynamoDB, usa o. `sns` `dynamodb`

- `region`è il codice AWS regionale in cui è stata creata l'attività di Step Functions o il tipo di macchina a stati, la funzione Lambda o qualsiasi altra AWS risorsa.
- `account`è l'Account AWSID in cui è stata definita la risorsa.
- `task_type` è il tipo di task da eseguire. Può essere uno dei seguenti valori:
  - `activity`— Un'attività.
  - `function`— Una funzione Lambda.
  - *`servicename`*— Il nome di un servizio connesso supportato (vediIntegrazioni ottimizzate per Step Functions).
- `name`è il nome della risorsa registrata (nome dell'attività, nome della funzione Lambda o azione dell'API del servizio).

> **Note**
>
> Step Functions non supporta il riferimento agli ARN tra partizioni o regioni. Ad esempio, non è `aws-cn` possibile richiamare attività nella `aws` partizione e viceversa.

Nelle sezioni seguenti vengono fornite ulteriori informazioni su ogni tipo di task.

Attività

Le attività rappresentano lavoratori (processi o thread), da te implementati e ospitati, che eseguono un task specifico. Sono supportati solo dai flussi di lavoro standard, non dai flussi di lavoro Express.

Gli ARN `Resource` di attività utilizzano la seguente sintassi.

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

> **Note**
>
> È necessario creare attività con Step Functions (utilizzando un'CreateActivityazione API o la console Step Functions) prima del loro primo utilizzo.

Per ulteriori informazioni sulla creazione di un'attività e l'implementazione di lavoratori, consultare Attività.

Funzioni Lambda

Le attività Lambda eseguono una funzione utilizzando. AWS Lambda Per specificare una funzione Lambda, usa l'ARN della funzione Lambda nel campo. `Resource`

A seconda del tipo di integrazione (integrazione ottimizzata o integrazione AWS SDK) utilizzato per specificare una funzione Lambda, la sintassi del campo della funzione Lambda varia. `Resource`

La seguente sintassi di `Resource` campo è un esempio di integrazione ottimizzata con una funzione Lambda.

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

La seguente sintassi di `Resource` campo è un esempio di integrazione AWS SDK con una funzione Lambda.

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

La seguente definizione di `Task` stato mostra un esempio di integrazione ottimizzata con una funzione Lambda denominata. *`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"
}
```

Una volta completata la funzione Lambda specificata nel `Resource` campo, il relativo output viene inviato allo stato identificato nel `Next` campo (» «). NextState

A supportata Servizio AWS

Quando si fa riferimento a una risorsa connessa, Step Functions richiama direttamente le azioni API di un servizio supportato. Specifica il servizio e l'operazione nel campo `Resource`.

Gli ARN `Resource` del servizio connesso utilizzano la seguente sintassi.

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

> **Note**
>
> Per creare una connessione sincrona a una risorsa connessa, aggiungi `.sync` alla voce *APIname* nell'ARN. Per ulteriori informazioni, consulta Uso di altri servizi.

Per esempio:

```
{
 "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
    }
  }
}
```

Campi relativi allo stato delle attività

Oltre ai campi di stato comuni, gli stati `Task` hanno i seguenti campi.

`Resource` (Obbligatorio)

Un URI, in particolare un ARN che identifica in modo univoco il task specifico da eseguire.

`Parameters` (facoltativo).

Utilizzato per passare le informazioni alle operazioni API delle risorse connesse. I parametri possono utilizzare una combinazione di JSON statico e JsonPath. Per ulteriori informazioni, consulta Passa i parametri a un'API di servizio.

`Credentials` (facoltativo).

Specifica un ruolo di destinazione che il ruolo di esecuzione della macchina a stati deve assumere prima di richiamare quello specificato. `Resource` In alternativa, puoi anche specificare un valore JSONPath o una funzione intrinseca che si risolve in un ruolo IAM ARN in fase di esecuzione in base all'input di esecuzione. Se specifichi un valore JSONPath, devi prefissarlo con la notazione. `$.`

Per esempi di utilizzo di questo campo nello stato, vedi. `Task` Esempi di campi relativi alle credenziali dello stato dell'attività Per un esempio di utilizzo di questo campo per accedere a una AWS risorsa multiaccount dalla macchina a stati, vediTutorial: Accesso alle risorse di più account AWS.

> **Note**
>
> Questo campo è supportato da chi utilizza le Tipi di attività funzioni Lambda e da un servizio supportato AWS.

`ResultPath` (facoltativo).

Specifica dove (nell'input) posizionare i risultati dell'esecuzione del task specificato in `Resource`. L'input viene quindi filtrato come specificato dal campo `OutputPath` (se presente) prima di essere utilizzato come output dello stato. Per ulteriori informazioni, consultare Elaborazione di input e output.

`ResultSelector` (facoltativo).

Passa una raccolta di coppie chiave-valore, in cui i valori sono statici o selezionati dal risultato. Per ulteriori informazioni, consulta ResultSelector.

`Retry` (facoltativo).

Un array di oggetti, denominati retrier, che definiscono una policy per i nuovi tentativi se si verificano errori di runtime nello stato. Per ulteriori informazioni, consulta Esempi di macchine a stati che utilizzano Retry e Catch.

`Catch` (facoltativo).

Un array di oggetti, denominati catcher, che definiscono uno stato di fallback. Questo stato viene eseguito se rileva errori di runtime e la relativa policy per i nuovi tentativi è esaurita o non è definita. Per ulteriori informazioni, consulta Stati di fallback.

`TimeoutSeconds` (facoltativo).

Speciifica il tempo massimo di esecuzione di un'attività o di un'attività prima che scada con l'States.Timeouterrore e abbia esito negativo. Il valore di timeout deve essere positivo, un numero intero diverso da zero. Il valore predefinito è 99999999.

Il conteggio dei timeout inizia dopo l'avvio di un'attività, ad esempio quando `LambdaFunctionStarted` gli eventi `ActivityStarted` o gli eventi vengono registrati nella cronologia degli eventi di esecuzione. Per le attività, il conteggio inizia quando `GetActivityTask` riceve un token e `ActivityStarted` viene registrato nella cronologia degli eventi di esecuzione.

All'avvio di un'attività, Step Functions attende una risposta positiva o negativa da parte dell'operatore dell'attività o dell'addetto all'attività entro la `TimeoutSeconds` durata specificata. Se l'operatore dell'attività o dell'attività non risponde entro questo periodo, Step Functions contrassegna l'esecuzione del flusso di lavoro come non riuscita.

`TimeoutSecondsPath` (facoltativo).

Se desideri fornire un valore di timeout in modo dinamico dall'input dello stato utilizzando un percorso di riferimento, usa. `TimeoutSecondsPath` Una volta risolto, il percorso di riferimento deve selezionare campi i cui valori sono numeri interi positivi.

> **Note**
>
> Uno `Task` stato non può includere entrambi `TimeoutSeconds` e`TimeoutSecondsPath`.

`HeartbeatSeconds` (facoltativo).

Determina la frequenza dei segnali del battito cardiaco che un Activity Worker invia durante l'esecuzione di un'attività. I battiti cardiaci indicano che un'attività è ancora in esecuzione e richiede più tempo per essere completata. I battiti cardiaci impediscono che un'attività o un'attività scada entro la durata prevista. `TimeoutSeconds`

`HeartbeatSeconds`deve essere un valore intero positivo, diverso da zero, inferiore al valore del campo. `TimeoutSeconds` Il valore predefinito è 99999999. Se trascorre più tempo dei secondi specificati tra i battiti cardiaci dell'operazione, lo stato dell'operazione ha esito negativo e viene generato un errore. States.Timeout

Per le attività, il conteggio inizia quando `GetActivityTask` riceve un token e `ActivityStarted` viene registrato nella cronologia degli eventi di esecuzione.

`HeartbeatSecondsPath` (facoltativo).

Se desideri fornire un valore del battito cardiaco in modo dinamico dall'input dello stato utilizzando un percorso di riferimento, usa. `HeartbeatSecondsPath` Una volta risolto, il percorso di riferimento deve selezionare campi i cui valori sono numeri interi positivi.

> **Note**
>
> Uno `Task` stato non può includere entrambi `HeartbeatSeconds` e`HeartbeatSecondsPath`.

Uno stato `Task` deve impostare il campo `End` su `true` se lo stato termina l'esecuzione oppure deve fornire uno stato nel campo `Next` che verrà eseguito al completamento dello stato `Task`.

Esempi di definizione dello stato dell'attività

Gli esempi seguenti mostrano come specificare la definizione dello stato dell'attività in base alle proprie esigenze.

- Specificazione dei timeout dello stato dell'attività e degli intervalli del battito cardiaco
  - Esempio di notifica di timeout statico e battito cardiaco
  - Esempio di notifica dinamica di timeout delle attività e battito cardiaco
- Utilizzo del campo Credenziali
  - Specificazione dell'ARN del ruolo IAM codificato
  - Specificare JSONPath come ARN del ruolo IAM
  - Specificazione di una funzione intrinseca come ARN del ruolo IAM

Timeout dello stato delle attività e intervalli di battito cardiaco

È buona norma impostare un valore di timeout e un intervallo di heartbeat per le attività di lunga durata. Questa operazione può essere eseguita specificando i valori di timeout e heartbeat o impostandoli dinamicamente.

Esempio di notifica di timeout statico e battito cardiaco

Al completamento di `HelloWorld`, verrà eseguito lo stato successivo (qui denominato `NextState`).

Se non viene completato entro 300 secondi o non invia notifiche heartbeat a intervalli di 60 secondi, questo task viene contrassegnato come `failed`.

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

Esempio di notifica dinamica di timeout delle attività e battito cardiaco

In questo esempio, al termine del AWS Glue processo, verrà eseguito lo stato successivo.

Se questa attività non viene completata entro l'intervallo impostato dinamicamente dal AWS Glue lavoro, l'attività viene contrassegnata come. `failed`

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

  "Next": "NextState"
}
```

Esempi di campi relativi alle credenziali dello stato dell'attività

Specificazione dell'ARN del ruolo IAM codificato

L'esempio seguente specifica un ruolo IAM di destinazione che il ruolo di esecuzione di una macchina a stati deve assumere per accedere a una funzione Lambda multiaccount denominata. `Echo` In questo esempio, l'ARN del ruolo di destinazione è specificato come valore codificato.

```
{
  "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
    }
  }
}
```

Specificare JSONPath come ARN del ruolo IAM

L'esempio seguente specifica un valore JSONPath, che verrà risolto in un ruolo IAM ARN in fase di esecuzione.

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

```
}
```

Specificazione di una funzione intrinseca come ARN del ruolo IAM

L'esempio seguente utilizza la funzione `States.Format`intrinseca, che si risolve in un ruolo IAM ARN in fase di esecuzione.

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

Attività

Le attività sono una AWS Step Functions funzionalità che consente di avere un'attività nella macchina statale in cui il lavoro viene eseguito da un lavoratore che può essere ospitata su Amazon Elastic Compute Cloud (Amazon EC2), Amazon Elastic Container Service (Amazon ECS), dispositivi mobili, praticamente ovunque.

Panoramica

In AWS Step Functions, le attività sono un modo per associare il codice in esecuzione da qualche parte (noto come lavoratore di attività) a un task specifico in una macchina a stati. Puoi creare un'attività utilizzando la console Step Functions o chiamando`CreateActivity`. Questo fornisce un Amazon Resource Name (ARN) per lo stato dell'attività. Utilizza questo ARN per effettuare il polling dello stato task per verificare la presenza di lavoro nel tuo lavoratore di attività.

> **Note**
>
> Le attività non hanno versioni e dovrebbero essere retrocompatibili. Se devi apportare una modifica incompatibile con le versioni precedenti a un'attività, crea una nuova attività in Step Functions utilizzando un nome univoco.

Un activity worker può essere un'applicazione in esecuzione su un'istanza Amazon EC2, una AWS Lambda funzione, un dispositivo mobile: qualsiasi applicazione in grado di stabilire una connessione HTTP, ospitata ovunque. Quando Step Functions raggiunge lo stato di attività, il flusso di lavoro attende che un operatore di attività effettui un sondaggio per un'attività. Un addetto alle attività effettua un sondaggio su Step Functions utilizzando `GetActivityTask` e inviando l'ARN per l'attività correlata. `GetActivityTask`restituisce una risposta che include `input` (una stringa di input JSON per l'attività) e un `taskToken`(un identificatore univoco per l'attività). Dopo aver completato il lavoro, il lavoratore di attività può fornire un report dell'esito positivo o negativo utilizzando `SendTaskSuccess` o `SendTaskFailure`. Queste due chiamate utilizzano il `taskToken` offerto da `GetActivityTask` per associare il risultato al task.

API relative ai task di attività

Step Functions fornisce API per creare ed elencare attività, richiedere un'attività e gestire il flusso della macchina statale in base ai risultati del lavoratore.

Di seguito sono elencate le API di Step Functions correlate alle attività:

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

> **Note**
>
> Il polling per i task di attività con `GetActivityTask` può risultare in una latenza in alcune implementazioni. Consultare Evita la latenza durante i sondaggi relativi alle attività.

Attesa del completamento di un task di attività

Configura il periodo di attesa di uno stato impostando `TimeoutSeconds` nella definizione del task. Per mantenere il task attivo e in attesa, invia periodicamente un heartbeat dal tuo lavoratore di attività utilizzando `SendTaskHeartbeat` entro l'intervallo di tempo configurato in `TimeoutSeconds`. Configurando una durata di timeout prolungata e inviando attivamente un battito cardiaco, un'attività in Step Functions può attendere fino a un anno per il completamento dell'esecuzione.

Se ad esempio hai bisogno di un flusso di lavoro che attenda l'esito di un processo lungo, esegui le operazioni descritte di seguito:

1. Crea un'attività tramite la console oppure utilizzando `CreateActivity`. Annota l'ARN dell'attività.
2. Fai riferimento all'ARN in uno stato task di attività nella definizione della tua macchina a stati e imposta `TimeoutSeconds`.
3. Implementa un lavoratore di attività che esegua il polling per ricercare un lavoro utilizzando `GetActivityTask` e facendo riferimento all'ARN dell'attività.
4. Utilizza `SendTaskHeartbeat` periodicamente entro l'intervallo di tempo impostato in `HeartbeatSeconds` nella definizione di task della tua macchina a stati per impedire il timeout del task.
5. Avvia un'esecuzione della tua macchina a stati.
6. Avvia il processo del lavoratore di attività.

L'esecuzione si ferma nello stato task di attività e attende che il tuo lavoratore di attività effettui il polling di un task. Una volta fornito un `taskToken` al tuo lavoratore di attività, il flusso di lavoro attenderà che `SendTaskSuccess` o `SendTaskFailure` fornisca uno stato. Se l'esecuzione non ottiene risultati o una chiamata `SendTaskHeartbeat` prima dell'intervallo di tempo configurato in `TimeoutSeconds`, l'esecuzione avrà esito negativo e la cronologia delle esecuzioni conterrà un evento `ExecutionTimedOut`.

Fasi successive

Per una panoramica dettagliata sulla creazione di una macchina a stati che utilizzi un lavoratore di attività, consulta:

- Creazione di una macchina a stati Activity utilizzando Step Functions
- Esempio di lavoratore di attività in Ruby

Esempio di lavoratore di attività in Ruby

Di seguito è riportato un esempio di lavoratore di attività che utilizza il kit AWS SDK for Ruby per mostrarti come sfruttare le best practice e implementare il tuo lavoratore di attività.

Il codice implementa un modello consumatore-produttore con un numero configurabile di thread per poller e lavoratori di attività. I thread di poller effettuano costantemente un long polling del task di attività. Una volta recuperato, un task di attività viene passato attraverso una coda di blocco delimitata in modo da essere raccolto dal thread di attività.

- Per ulteriori informazioni suAWS SDK for Ruby, consulta l'AWS SDK for RubyAPI Reference.
- Per scaricare questo codice e le risorse correlate, consulta il repository step-functions-ruby-activity-worker su. GitHub

Il seguente codice Ruby è il punto di ingresso principale per questo esempio di lavoratore di attività Ruby.

```
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
```

Il codice include impostazioni predefinite che puoi modificare in modo da fare riferimento alla tua attività e adattarla alla tua implementazione specifica. Questo codice prende come input la logica di implementazione effettiva, consente di fare riferimento alla tua attività e alle tue credenziali specifiche

e ti permette di configurare il numero di thread e il ritardo dell'heartbeat. Per ulteriori informazioni e per scaricare il codice, vedere Step Functions Ruby Activity Worker.

Elemento	Descrizione
`require_relative`	Percorso relativo per il seguente esempio di codice di lavoratore di attività.
`region`	Regione AWS della tua attività.
`workers_count`	Il numero di thread per il tuo lavoratore di attività. Per la maggior parte delle implement azioni, dovrebbero essere sufficienti tra i 10 e i 20 thread. Più tempo l'attività impiega a elaborare, più thread potrebbero servire. Per fare una stima, moltiplica il numero di attività di elaborazione al secondo in base al 99esimo della latenza di elaborazione di attività, in secondi.
`pollers_count`	Il numero di thread per i tuoi poller. Per la maggior parte delle implementazioni, dovrebber o essere sufficienti tra i 10 e i 20 thread.
`heartbeat_delay`	Il ritardo tra heartbeat espresso in secondi.
`input`	Logica di implementazione della tua attività.

Di seguito è riportato il lavoratore di attività Ruby, con riferimento `../lib/step_functions/activity` nel tuo codice.

```
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
```

Choice

Uno `Choice` state (`"Type": "Choice"`) aggiunge la logica condizionale a una macchina a stati.

Oltre alla maggior parte dei campi di stato comuni, `Choice` gli stati contengono i seguenti campi aggiuntivi.

`Choices` (Obbligatorio)

Un array di regole Choice che determina lo stato successivo della macchina a stati. Si utilizza un operatore di confronto in una regola di scelta per confrontare una variabile di input con un valore specifico. Ad esempio, utilizzando le regole di scelta è possibile confrontare se una variabile di input è maggiore o minore di 100.

Quando uno `Choice` stato viene eseguito, valuta ogni regola di scelta come vera o falsa. In base al risultato di questa valutazione, Step Functions passa allo stato successivo del flusso di lavoro.

È necessario definire almeno una regola nello `Choice` stato.

`Default` (Facoltativo, Consigliato)

Il nome dello stato al quale passare se nessuna delle transizioni in `Choices` viene considerata.

> **Important**
>
> Gli stati `Choice` non supportano il campo `End`. Inoltre, utilizzano `Next` solo nel relativo campo `Choices`.

> **Tip**
>
> Per implementare un esempio di flusso di lavoro che utilizza `Choice` uno stato nel tuoAccount AWS, vedi Modulo 5 - Choice State e Map State of The AWS Step Functions Workshop.

Regole Choice

Uno `Choice` stato deve avere un `Choices` campo il cui valore è un array non vuoto. Ogni elemento di questo array è un oggetto chiamato Choice Rule, che contiene quanto segue:

- Un confronto: due campi che specificano una variabile di input da confrontare, il tipo di confronto e il valore con cui confrontare la variabile. Le regole di scelta supportano il confronto tra due variabili. All'interno di una regola di scelta, il valore della variabile può essere confrontato con un altro valore dell'input di stato aggiungendolo `Path` al nome degli operatori di confronto supportati. I valori `Variable` e i campi Percorso in un confronto devono essere Percorsi di riferimento validi.
- Un **`Next`**campo: il valore di questo campo deve corrispondere al nome di uno stato nella macchina a stati.

L'esempio seguente verifica se il valore numerico è uguale a `1`.

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

L'esempio seguente verifica se la stringa è uguale a `MyString`.

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

L'esempio seguente verifica se la stringa è maggiore di `MyStringABC`.

```
{
```

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

L'esempio seguente verifica se la stringa è nulla.

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

L'esempio seguente mostra come la StringEquals regola viene valutata solo quando `$.keyThatMightNotExist` esiste a causa della precedente `IsPresent` Choice Rule.

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

L'esempio seguente verifica se un pattern con un carattere jolly corrisponde.

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

L'esempio seguente verifica se il timestamp è uguale a `2001-01-01T12:00:00Z`.

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

L'esempio seguente confronta una variabile con un altro valore dell'input di stato.

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

Step Functions esamina ciascuna delle regole di scelta nell'ordine elencato nel `Choices` campo. Quindi passa allo stato specificato nel campo `Next` della prima regola Choice in cui la variabile corrisponde al valore in base all'operatore di confronto.

Sono supportati i seguenti operatori di confronto:

- `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`

Per ognuno di questi operatori, il valore corrispondente deve essere del tipo appropriato: stringa, numero, booleano o timestamp. Step Functions non tenta di far corrispondere un campo numerico a un valore di stringa. Tuttavia, poiché i campi timestamp sono logicamente delle stringhe, è possibile che un campo considerato come timestamp possa corrispondere a un comparatore `StringEquals`.

> **Note**
>
> Per l'interoperabilità, non supporre che i confronti numerici funzionino con valori al di fuori della grandezza o della precisione che il tipo di dati IEEE 754-2008`binary64` rappresenta. In particolare, gli interi che non rientrano nell'intervallo `[-`2^{53}`+1, `2^{53}`-1]` potrebbero non essere confrontati come previsto.
> I timestamp (ad esempio, `2016-08-18T17:33:00Z`) devono essere conformi al profilo RFC3339 ISO 8601, con ulteriori limitazioni:
>
> - Una `T` maiuscola deve separare la parte della data dalla parte dell'ora.
> - Una `Z` maiuscola indica che al momento non è disponibile l'offset del fuso orario numerico.
>
> Per comprendere il funzionamento dei confronti di stringhe, consultare la documentazione di `compareTo`.
> I valori degli operatori `And` e `Or` devono essere array non vuoti di regole Choice che non devono contenere campi `Next`. Inoltre, il valore di un operatore `Not` deve essere una singola regola Choice che non deve contenere campi `Next`.
> Puoi creare regole Choice nidificate complesse utilizzando `And`, `Not` e `Or`. Tuttavia, il campo `Next` può essere visualizzato solo in una regola Choice di livello superiore.
> Il confronto tra stringhe e modelli con uno o più caratteri jolly («*») può essere eseguito con l'operatore di StringMatches confronto. Il carattere jolly viene evaso utilizzando lo standard. `\\ (Ex: “\\*”)` Nessun carattere diverso da «*» ha un significato speciale durante l'abbinamento.

Esempio di stato Choice

Di seguito è riportato un esempio di stato `Choice` e altri stati a cui è possibile passare.

> **Note**
>
> Devi specificare il campo `$.type`. Se l'input dello stato non contiene il campo `$.type`, l'esecuzione non riesce e viene visualizzato un messaggio di errore nella cronologia delle esecuzioni. È possibile specificare solo una stringa nel `StringEquals` campo che corrisponde a un valore letterale. Ad esempio, `"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!"
}
```

In questo esempio, la macchina a stati inizia con il seguente valore di input.

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

Step Functions: transizioni allo `ValueInTwenties` stato, in base al `value` campo.

Se non vi sono corrispondenze con il campo `Choices` dello stato `Choice`, viene eseguito lo stato fornito nel campo `Default`. Se lo stato `Default` non è specificato, l'esecuzione non riesce con un errore.

Attendi

Uno stato `Wait` (`"Type": "Wait"`) impedisce il funzionamento della macchina a stati per un determinato periodo di tempo. Puoi scegliere tra un intervallo di tempo relativo, specificato in

secondi a partire dal momento in cui inizia lo stato, oppure un'ora di fine assoluta, specificata come timestamp.

Oltre ai campi di stato comuni, gli stati `Wait` hanno uno dei seguenti campi.

`Seconds`

Un tempo di attesa in secondi prima dell'inizio dello stato specificato nel campo `Next`. È necessario specificare l'ora come valore intero positivo compreso tra 0 e 99999999.

`Timestamp`

Un tempo di attesa assoluto fino all'inizio dello stato specificato nel campo `Next`.

I timestamp devono essere conformi al profilo RFC3339 della norma ISO 8601, con ulteriori restrizioni secondo cui una T maiuscola deve separare la data e l'ora e una Z maiuscola deve indicare che un offset di fuso orario non è presente, ad esempio, `2024-08-18T17:33:00Z`.

> **Note**
>
> Attualmente, se si specifica il tempo di attesa come timestamp, Step Functions considera il valore temporale fino a secondi e tronca i millisecondi.

`SecondsPath`

Un tempo di attesa in secondi prima dell'inizio dello stato indicato nel campo `Next`, specificato utilizzando un percorso dai dati di input dello stato.

È necessario specificare un valore intero per questo campo.

`TimestampPath`

Un tempo di attesa assoluto fino all'inizio dello stato specificato nel campo `Next` utilizzando un percorso dai dati di input dello stato.

> **Note**
>
> Devi specificare esattamente `Seconds`, `Timestamp`, `SecondsPath` o `TimestampPath`. Inoltre, il tempo di attesa massimo che è possibile specificare per i flussi di lavoro Standard e i flussi di lavoro Express è rispettivamente di un anno e cinque minuti.

Esempi di stato Wait

Il seguente stato `Wait` introduce un ritardo di 10 secondi in una macchina a stati.

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

Nel prossimo esempio, lo `Wait` stato attende fino a un'ora assoluta: il 14 marzo 2024 alle 01:59 UTC.

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

Non è necessario effettuare l'hard coding del tempo di attesa. Ad esempio, con i seguenti dati di input:

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

Puoi selezionare il valore di "expirydate" dall'input utilizzando un percorso di riferimento per selezionarlo dai dati di input.

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

Succeed

Uno stato `Succeed` (`"Type": "Succeed"`) arresta un'esecuzione. Lo stato `Succeed` è un target utile per i rami dello stato `Choice` la cui unica funzione è di arrestare l'esecuzione.

Poiché sono terminali, gli stati `Succeed` non hanno il campo `Next` e non hanno bisogno di un campo `End`, come mostrato nell'esempio seguente.

```
"SuccessState": {
  "Type": "Succeed"
}
```

Fail

UN`Fail`stato (`"Type": "Fail"`) interrompe l'esecuzione della macchina a stati e la contrassegna come fallita, a meno che non venga intercettata da un`Catch`bloccare.

Lo stato `Fail` autorizza soltanto l'utilizzo dei campi `Type` e `Comment` dal set di campi di stato comuni. Inoltre, lo stato `Fail` consente i seguenti campi.

`Cause` (facoltativo).

Una stringa personalizzata che descrive la causa dell'errore. È possibile specificare questo campo per scopi operativi o diagnostici.

`CausePath` (facoltativo).

Se si desidera fornire una descrizione dettagliata della causa dell'errore in modo dinamico a partire dallo stato immesso utilizzando unpercorso di riferimento, usa`CausePath`. Una volta risolto, il percorso di riferimento deve selezionare un campo che contiene un valore di stringa.

È inoltre possibile specificare`CausePath`utilizzando unfunzione intrinsecache restituisce una stringa. Questi elementi intrinseci sono:Stati.Formato,States.JsonToString,States.ArrayGetItem,States.Base64Encode,States.Base64Decod Hash, eStates.UUID.

> **Important**
>
> - È possibile specificare`Cause`o`CausePath`, ma non entrambi nella definizione dello stato Fail.
> - Per una sicurezza ottimale, è consigliabile non memorizzare dati di sicurezza, è consigliabile rimuovere dati di sistema dalla descrizione della causa.

`Error` (facoltativo).

Un nome di errore che è possibile fornire per eseguire la gestione degli errori utilizzandoRiprovaoCatturacampi. È inoltre possibile fornire un nome di errore per scopi operativi o diagnostici.

`ErrorPath` (facoltativo).

Se si desidera fornire un nome per l'errore in modo dinamico a partire dall'input dello stato utilizzando unpercorso di riferimento, usa`ErrorPath`. Una volta risolto, il percorso di riferimento deve selezionare un campo che contiene un valore di stringa.

È inoltre possibile specificare`ErrorPath`utilizzando unfunzione intrinsecache restituisce una stringa. Questi elementi intrinseci sono:Stati.Formato,States.JsonToString,States.ArrayGetItem,States.Base64Encode,States.Base64Decod Hash, eStates.UUID.

> **Important**
>
> - È possibile specificare`Error`o`ErrorPath`, ma non entrambi nella definizione dello stato Fail.
> - Per una sicurezza ottimale, è consigliabile rimuovere dati di sicurezza dal nome dell'errore.

Poiché gli stati `Fail` escono sempre dalla macchina a stati, non hanno campi `Next` e non richiedono un campo `End`.

Esempi di definizione dello stato di sicurezza

Il seguente esempio di definizione dello stato di errore specifica lo stato statico`Error`e`Cause`valori di campo.

```
"FailState": {
  "Type": "Fail",
  "Cause": "Invalid response.",
  "Error": "ErrorA"
}
```

Il seguente esempio di definizione dello stato di errore utilizza percorsi di riferimento in modo dinamico per risolvere il`Error`e`Cause`valori di campo.

```
"FailState": {
  "Type": "Fail",
  "CausePath": "$.Cause",
  "ErrorPath": "$.Error"
}
```

Il seguente esempio di definizione dello stato Fail utilizzaStates.Formatfunzione intrinseca per specificare il`Error`e`Cause`valori di campo in modo dinamico.

```
"FailState": {
  "Type": "Fail",
  "CausePath": "States.Format('This is a custom error message for {}, caused by {}.',
 $.Error, $.Cause)",
  "ErrorPath": "States.Format('{}', $.Error)"
}
```

Parallel

Lo `Parallel` stato (`"Type": "Parallel"`) può essere usato per aggiungere rami di esecuzione separati nella macchina a stati.

Oltre ai campi di stato comuni, gli stati `Parallel` includono i seguenti campi aggiuntivi.

`Branches` (Obbligatorio)

Un array di oggetti che specificano le macchine a stati da eseguire in parallelo. Ognuno di questi oggetti deve avere campi denominati `States` e `StartAt`, i cui significati sono esattamente uguali a quelli nel livello superiore di una macchina a stati.

`ResultPath` (facoltativo).

Specifica dove (nell'input) posizionare l'output dei rami. L'input viene quindi filtrato come specificato dal campo `OutputPath` (se presente) prima di essere utilizzato come output dello stato. Per ulteriori informazioni, consultare Elaborazione di input e output.

`ResultSelector` (facoltativo).

Passa una raccolta di coppie chiave-valore, in cui i valori sono statici o selezionati dal risultato. Per ulteriori informazioni, consulta ResultSelector.

`Retry` (facoltativo).

Un array di oggetti, denominati retrier, che definiscono una policy per i nuovi tentativi nel caso in cui si verifichino errori di runtime nello stato. Per ulteriori informazioni, consulta Esempi di macchine a stati che utilizzano Retry e Catch.

`Catch` (facoltativo).

Un array di oggetti, denominati catcher, che definiscono uno stato di fallback che viene eseguito se si verificano errori di runtime nello stato e la relativa policy per i nuovi tentativi risulta esaurita o non definita. Per ulteriori informazioni, consulta Stati di fallback.

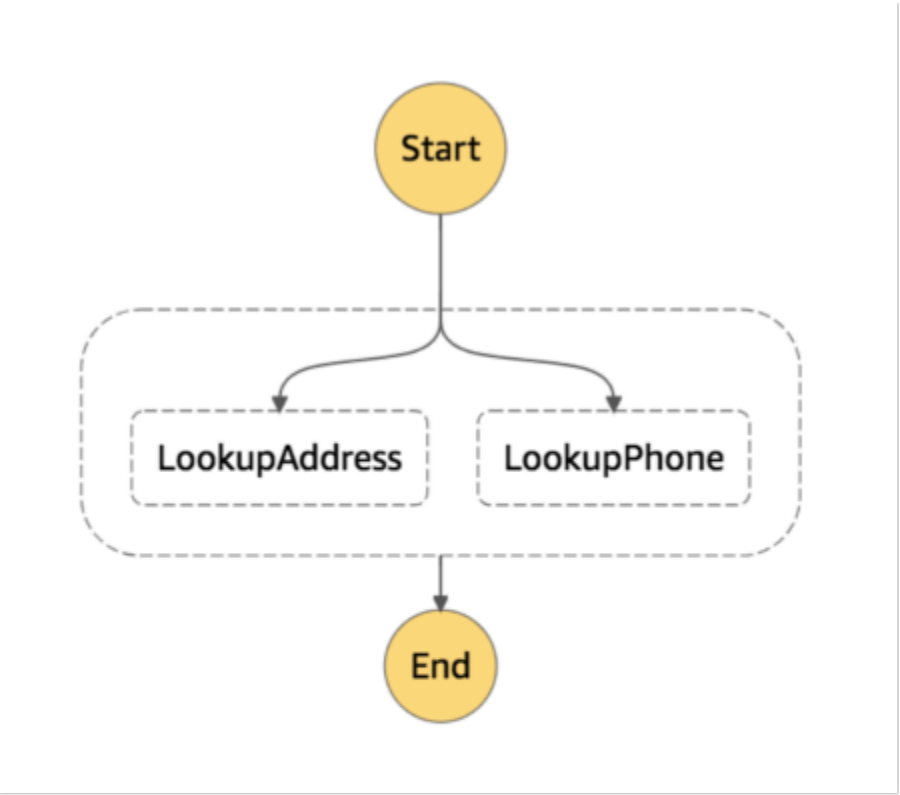
Uno `Parallel` stato fa sì AWS Step Functions che ogni ramo, a partire dallo stato indicato nel `StartAt` campo di quel ramo, venga eseguito il più contemporaneamente possibile, e attendere che tutti i rami terminino (raggiungano uno stato terminale) prima di elaborare il campo `Parallel` dello `Next` stato.

Esempio di stato Parallel

```
{
  "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
          }
        }
      }
    ]
  }
 }
}
```

Nel presente esempio, i rami `LookupAddress` e `LookupPhone` sono eseguiti in parallelo. Ecco come appare il flusso di lavoro visivo nella console Step Functions.



Ogni ramo deve essere autonomo. Uno stato in un ramo di uno stato `Parallel` non deve avere un campo `Next` che utilizza come target un campo al di fuori di quel ramo, né qualsiasi altro stato al di fuori della transizione di ramo in quel ramo.

Elaborazione di input e output dello stato Parallel

Uno stato `Parallel` fornisce a ogni ramo una copia dei relativi dati di input (soggetti a modifica mediante il campo `InputPath`). Genera un output che è un array con un elemento per ogni ramo, contenente l'output di quel ramo. Tutti gli elementi non devono essere obbligatoriamente dello stesso tipo. L'array di output può essere inserito nei dati di input (e il tutto inviato come output dello stato `Parallel`) utilizzando un campo `ResultPath` nel modo consueto (vedi Elaborazione di input e 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
            }
          }
        }
      ]
```

```
    }
  }
}
```

Se allo stato `FunWithMath` viene assegnato l'array `[3, 2]` come input, gli stati `Add` e `Subtract` ricevono quell'array come input. L'output di `Add` and `Subtract` tasks sarebbe la somma e la differenza tra gli elementi 3 e 2 dell'array, ovvero `5` e`1`, mentre l'output dello `Parallel` stato sarebbe un array.

```
[ 5, 1 ]
```

> **Tip**
>
> Se lo stato Parallel o Map che usi nelle tue macchine a stati restituisce un array di array, puoi trasformarli in una matrice piatta con il ResultSelector campo. Per ulteriori informazioni, consulta Appiattimento di una serie di array.

Gestione errori

Se un ramo ha esito negativo, a causa di un errore non gestito o a seguito della transizione a uno stato `Fail`, l'intero stato `Parallel` viene considerato come non riuscito e tutti i relativi rami vengono arrestati. Se l'errore non viene gestito dallo `Parallel` stato stesso, Step Functions interrompe l'esecuzione con un errore.

> **Note**
>
> Quando si verifica un errore in uno stato parallelo, le funzioni Lambda richiamate continuano a essere eseguite e gli activity worker che elaborano un task token non vengono interrotti.
>
> - Per interrompere le attività di lunga durata, utilizza gli heartbeats per rilevare se la succursale è stata interrotta da Step Functions e interrompi i lavoratori che stanno elaborando le attività. La chiamata di `SendTaskHeartbeat`, `SendTaskSuccess` o `SendTaskFailure` genererà un errore se lo stato ha esito negativo. Consulta la selezione relativa agli Errori di heartbeat.
> - L'esecuzione delle funzioni Lambda non può essere interrotta. Se hai implementato un fallback, usa uno `Wait` stato in modo che il lavoro di pulizia avvenga al termine della funzione Lambda.

Eseguire la mappatura

Utilizzate lo `Map` stato per eseguire una serie di passaggi del flusso di lavoro per ogni elemento di un set di dati. Le iterazioni `Map` dello stato vengono eseguite in parallelo, il che rende possibile elaborare rapidamente un set di dati. `Map`gli stati possono utilizzare diversi tipi di input, tra cui un array JSON, un elenco di oggetti Amazon S3 o un file CSV.

Step Functions offre due tipi di modalità di elaborazione per l'utilizzo dello `Map` stato nei flussi di lavoro: modalità Inline e modalità Distributed.

Per informazioni su queste modalità e su come utilizzare `Map` lo stato in entrambe le modalità, consultate i seguenti argomenti:

- Modalità di elaborazione dello stato della mappa
  - Utilizzo dello stato della mappa in modalità Inline
  - Utilizzo dello stato della mappa in modalità distribuita

> **Tip**
>
> Per distribuire un esempio di flusso di lavoro che utilizza `Map` uno stato per il tuoAccount AWS, consulta il Modulo 5 - Choice State e Map State of The AWS Step Functions Workshop.

Modalità di elaborazione dello stato della mappa

Step Functions fornisce le seguenti modalità di elaborazione per lo `Map` stato, a seconda di come si desidera elaborare gli elementi in un set di dati.

- Inline: modalità a concorrenza limitata. In questa modalità, ogni iterazione dello `Map` stato viene eseguita nel contesto del flusso di lavoro che contiene lo stato. `Map` Step Functions aggiunge la cronologia di esecuzione di queste iterazioni alla cronologia di esecuzione del flusso di lavoro principale. Per impostazione predefinita, `Map` gli stati vengono eseguiti in modalità Inline.

  In questa modalità, lo `Map` stato accetta solo un array JSON come input. Inoltre, questa modalità supporta fino a 40 iterazioni simultanee.

  Per ulteriori informazioni, consulta Utilizzo dello stato della mappa in modalità Inline.

- Distribuita: modalità ad alta concorrenza. In questa modalità, lo `Map` stato esegue ogni iterazione come esecuzione di workflow secondario, il che consente un'elevata concorrenza di un massimo di 10.000 esecuzioni parallele di workflow secondari. Ogni esecuzione del flusso di lavoro secondario ha una propria cronologia di esecuzione separata da quella del flusso di lavoro principale.

  In questa modalità, lo `Map` stato può accettare come input un array JSON o un'origine dati Amazon S3, ad esempio un file CSV.

  Per ulteriori informazioni, consulta Utilizzo dello stato della mappa in modalità distribuita.

La modalità da utilizzare dipende da come si desidera elaborare gli elementi in un set di dati. Utilizzate lo `Map` stato in modalità Inline se la cronologia di esecuzione del flusso di lavoro non supera le 25.000 voci o se non sono necessarie più di 40 iterazioni simultanee.

Usa lo `Map` stato in modalità Distribuita quando devi orchestrare carichi di lavoro paralleli su larga scala che soddisfano qualsiasi combinazione delle seguenti condizioni:

- La dimensione del set di dati supera i 256 KB.
- La cronologia degli eventi di esecuzione del flusso di lavoro supera le 25.000 voci.
- È necessaria una concorrenza di più di 40 iterazioni parallele.

Argomenti

- Differenze tra modalità in linea e modalità distribuita
- Utilizzo dello stato della mappa in modalità Inline
- Utilizzo dello stato della mappa in modalità distribuita per orchestrare carichi di lavoro paralleli su larga scala

Differenze tra modalità in linea e modalità distribuita

La tabella seguente evidenzia le differenze tra le modalità Inline e Distributed.

Modalità in linea	Modalità distribuita
Supported data sources	
Accetta come input un array JSON passato da una fase precedente del flusso di lavoro.	Accetta le seguenti fonti di dati come input:

Modalità in linea	Modalità distribuita
	• Array JSON passato da un passaggio precedente del flusso di lavoro • File JSON in un bucket Amazon S3 che contiene un array • File CSV in un bucket Amazon S3 • Elenco di oggetti Amazon S3 • Inventario Amazon S3
Map iterations	
In questa modalità, ogni iterazione dello `Map` stato viene eseguita nel contesto del flusso di lavoro che contiene lo `Map` stato. Step Functions aggiunge la cronologia di esecuzione di queste iterazioni alla cronologia di esecuzion e del flusso di lavoro principale.	In questa modalità, lo `Map` stato esegue ogni iterazione come esecuzione di workflow secondario, il che consente un'elevata concorrenza di un massimo di 10.000 esecuzioni parallele di workflow secondari. Ogni esecuzione del flusso di lavoro secondari o ha una propria cronologia di esecuzione separata da quella del flusso di lavoro principal e.
Maximum concurrency for parallel iterations	
Consente di eseguire fino a 40 iterazioni il più contemporaneamente possibile.	Consente di eseguire fino a 10.000 esecuzion i parallele di flussi di lavoro secondari per elaborare milioni di elementi di dati contempor aneamente.
Input payload and event history sizes	

Modalità in linea	Modalità distribuita
Implica un limite di 256 KB alla dimensione del payload di input e 25.000 voci nella cronologia degli eventi di esecuzione.	Consente di superare la limitazione delle dimensioni del payload perché `Map` lo stato può leggere l'input direttamente dalle fonti di dati Amazon S3. In questa modalità, puoi anche superare i limiti della cronologia di esecuzione, poiché le esecuzioni dei flussi di lavoro secondari avviate dallo `Map` stato mantengono le proprie cronologie di esecuzione separate dalla cronologia di esecuzione del flusso di lavoro principale.
Monitoring and observability	
È possibile rivedere la cronologia di esecuzione del flusso di lavoro dalla console o richiamando l'azione dell'`GetExecutionHistory` API. È inoltre possibile visualizzare la cronologia delle esecuzioni tramite CloudWatch X-Ray.	Quando si esegue uno `Map` stato in modalità Distribuita, Step Functions crea una risorsa Map Run. Un Map Run si riferisce a un insieme di esecuzioni di workflow secondarie avviate da uno stato di Distributed Map. È possibile visualizzare un Map Run nella console Step Functions. Puoi anche richiamare l'azione `DescribeMapRun` API. Un Map Run emette anche metriche per. CloudWatch Per ulteriori informazioni, consulta Esame dell'esecuzione dello stato di una mappa distribuita in esecuzione.

Utilizzo dello stato della mappa in modalità Inline

Per impostazione predefinita, `Map` gli stati vengono eseguiti in modalità Inline. In modalità Inline, lo stato della mappa accetta solo un array JSON come input. Riceve questo array da una fase precedente del flusso di lavoro. In questa modalità, ogni iterazione dello `Map` stato viene eseguita

nel contesto del flusso di lavoro che contiene lo `Map` stato. Step Functions aggiunge la cronologia di esecuzione di queste iterazioni alla cronologia di esecuzione del flusso di lavoro principale.

In questa modalità, lo `Map` stato supporta fino a 40 iterazioni simultanee.

Uno `Map` stato impostato su Inline è noto come stato Inline Map. Utilizza lo `Map` stato in modalità Inline se la cronologia di esecuzione del flusso di lavoro non supera le 25.000 voci o se non sono necessarie più di 40 iterazioni simultanee.

Per un'introduzione all'uso dello stato della mappa in linea, consulta il tutorial. Ripeti un'azione utilizzando lo stato della mappa in linea

Indice

- Concetti chiave in questo argomento
- Campi di stato della mappa in linea
- Campi obsoleti
- Esempio di stato della mappa in linea
- Esempio di stato della mappa in linea con ItemSelector
- Elaborazione di input e Map output in stato di linea

Concetti chiave in questo argomento

Modalità in linea

Una modalità di concorrenza limitata dello stato. `Map` In questa modalità, ogni iterazione dello `Map` stato viene eseguita nel contesto del flusso di lavoro che contiene lo stato. `Map` Step Functions aggiunge la cronologia di esecuzione di queste iterazioni alla cronologia di esecuzione del flusso di lavoro principale. `Map`gli stati vengono eseguiti in modalità Inline per impostazione predefinita.

Questa modalità accetta solo un array JSON come input e supporta fino a 40 iterazioni simultanee.

Stato della mappa in linea

Uno `Map` stato impostato sulla modalità Inline.

Mappa il flusso di lavoro

L'insieme di passaggi eseguiti dallo `Map` stato per ogni iterazione.

Iterazione dello stato della mappa

Una ripetizione del flusso di lavoro definito all'interno dello `Map` stato.

Campi di stato della mappa in linea

Per utilizzare lo stato Inline Map nei flussi di lavoro, specifica uno o più di questi campi. Specificate questi campi oltre ai campi di stato comuni.

`Type` (Obbligatorio)

Imposta il tipo di stato, ad esempio`Map`.

`ItemProcessor` (Obbligatorio)

Contiene i seguenti oggetti JSON che specificano la modalità e la definizione di elaborazione `Map` dello stato.

La definizione contiene l'insieme di passaggi da ripetere per l'elaborazione di ogni elemento dell'array.

- `ProcessorConfig`— Un oggetto JSON opzionale che specifica la modalità di elaborazione per lo `Map` stato. Questo oggetto contiene il sottocampo`Mode`. Il valore predefinito di questo campo è`INLINE`, che utilizza `Map` lo stato in modalità Inline.

  In questa modalità, l'errore di qualsiasi iterazione causa il fallimento dello `Map` stato. Tutte le iterazioni si interrompono quando lo `Map` stato fallisce.

- `StartAt`— specifica una stringa che indica il primo stato di un flusso di lavoro. Questa stringa fa distinzione tra maiuscole e minuscole e deve corrispondere al nome di uno degli oggetti di stato. Questo stato viene eseguito per primo per ogni elemento del set di dati. Qualsiasi input di esecuzione fornito allo `Map` stato passa per primo allo `StartAt` stato.
- `States`— Un oggetto JSON contenente un set di stati delimitato da virgole. In questo oggetto, si definisce il. Map workflow

> **Note**
>
> - Gli stati all'interno del `ItemProcessor` campo possono solo passare l'uno dall'altro. Nessuno stato esterno al `ItemProcessor` campo può passare a uno stato al suo interno.

- Il `ItemProcessor` campo sostituisce il campo ora obsoleto`Iterator`. Sebbene sia possibile continuare a includere `Map` gli stati che utilizzano il `Iterator` campo, si consiglia vivamente di sostituirlo con. `ItemProcessor`

  Step Functions Local attualmente non supporta il `ItemProcessor` campo. Ti consigliamo di utilizzare il `Iterator` campo con Step Functions Local.

`ItemsPath` (facoltativo).

Specificate un percorso di riferimento utilizzando la JsonPathsintassi. Questo percorso seleziona il nodo JSON che contiene l'array di elementi all'interno dell'input di stato. Per ulteriori informazioni, consulta ItemsPath.

`ItemSelector` (facoltativo).

Sostituisce i valori degli elementi dell'array di input prima che vengano passati a ciascuna iterazione di stato. `Map`

In questo campo, si specifica un JSON valido che contiene una raccolta di coppie chiave-valore. Queste coppie possono contenere uno dei seguenti elementi:

- Valori statici definiti nella definizione della macchina a stati.
- Valori selezionati dall'input dello stato utilizzando un percorso.
- Valori a cui si accede dall'oggetto contestuale.

Per ulteriori informazioni, consulta ItemSelector.

Il `ItemSelector` campo sostituisce il campo ora obsoleto`Parameters`. Sebbene sia possibile continuare a includere `Map` gli stati che utilizzano il `Parameters` campo, si consiglia vivamente di sostituirlo con. `ItemSelector`

`MaxConcurrency` (facoltativo).

Specifica un valore intero che fornisce il limite superiore del numero di iterazioni di `Map` stato che possono essere eseguite in parallelo. Ad esempio, un `MaxConcurrency` valore pari a 10 limita lo `Map` stato a 10 iterazioni simultanee eseguite contemporaneamente.

> **Note**
>
> Le iterazioni simultanee possono essere limitate. Quando ciò si verifica, alcune iterazioni non inizieranno fino al completamento delle iterazioni precedenti. La probabilità che ciò si verifichi aumenta quando l'array di input contiene più di 40 elementi.

Per ottenere una maggiore concorrenza, considera. Utilizzo dello stato della mappa in modalità distribuita

Il valore predefinito è`0`, che non pone limiti alla concorrenza. Step Functions richiama le iterazioni il più simultaneamente possibile.

`MaxConcurrency`Il valore di `1` invoca una volta per ogni elemento dell'`ItemProcessor`array. Gli elementi dell'array vengono elaborati nell'ordine in cui appaiono nell'input. Step Functions non avvia una nuova iterazione finché non completa l'iterazione precedente.

`MaxConcurrencyPath` (facoltativo).

Se desideri fornire un valore di concorrenza massimo in modo dinamico dall'input dello stato utilizzando un percorso di riferimento, usa. `MaxConcurrencyPath` Una volta risolto, il percorso di riferimento deve selezionare un campo il cui valore è un numero intero non negativo.

> **Note**
>
> Uno `Map` stato non può includere entrambi `MaxConcurrency` e. `MaxConcurrencyPath`

`ResultPath` (facoltativo).

Speciifica dove, nell'input, deve essere memorizzato l'output delle iterazioni dello `Map` stato. Lo stato Mappa filtra quindi l'input come specificato dal `OutputPath`campo, se specificato. Quindi, utilizza l'input filtrato come output dello stato. Per ulteriori informazioni, consultare Elaborazione di input e output.

`ResultSelector` (facoltativo).

Passa una raccolta di coppie chiave-valore, in cui i valori sono statici o selezionati dal risultato. Per ulteriori informazioni, consulta ResultSelector.

> **Tip**
>
> Se lo stato Parallel o Map che usi nelle tue macchine a stati restituisce un array di array, puoi trasformarli in un array flat con il ResultSelector campo. Per ulteriori informazioni, consulta Appiattimento di una serie di array.

`Retry` (facoltativo).

Una serie di oggetti, denominati Retriers, che definiscono una politica di nuovi tentativi. Gli stati utilizzano una politica di nuovo tentativo quando riscontrano errori di runtime. Per ulteriori informazioni, consulta Esempi di macchine a stati che utilizzano Retry e Catch.

> **Note**
>
> Se definisci Retriers per lo stato Inline Map, la politica di riprova si applica a tutte le iterazioni di `Map` stato, anziché solo alle iterazioni fallite. Ad esempio, `Map` lo stato contiene due iterazioni riuscite e un'iterazione fallita. Se hai definito il `Retry` campo per `Map` lo stato, la politica di riprova si applica a tutte e tre le iterazioni di `Map` stato anziché solo all'iterazione fallita.

`Catch` (facoltativo).

Un array di oggetti, denominati catcher, che definiscono uno stato di fallback. Gli stati eseguono un catcher se riscontrano errori di runtime e non dispongono di una politica di ripetizione dei tentativi oppure la politica di ripetizione dei tentativi è esaurita. Per ulteriori informazioni, consulta Stati di fallback.

Campi obsoleti

> **Note**
>
> Sebbene sia possibile continuare a includere `Map` gli stati che utilizzano i seguenti campi, si consiglia vivamente di sostituirli `Iterator` con e con`ItemProcessor`. `Parameters` `ItemSelector`

`Iterator`

Specifica un oggetto JSON che definisce una serie di passaggi che elaborano ogni elemento dell'array.

`Parameters`

Specifica una raccolta di coppie chiave-valore, in cui i valori possono contenere uno dei seguenti elementi:

- Valori statici definiti nella definizione della macchina a stati.
- Valori selezionati dall'input utilizzando un percorso.

Esempio di stato della mappa in linea

Considerate i seguenti dati di input per uno `Map` stato in esecuzione in modalità 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 }
    ]
  }
}
```

Dato l'input precedente, lo `Map` stato dell'esempio seguente richiama una AWS Lambda funzione denominata `ship-val` una volta per ogni elemento dell'array nel campo. `shipped`

```
"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"
}
```

Ogni iterazione dello `Map` stato invia un elemento dell'array, selezionato con il `ItemsPath`campo, come input per la funzione `ship-val` Lambda. I seguenti valori sono un esempio di input che lo `Map` stato invia a una chiamata della funzione Lambda:

```
{
  "prod": "R31",
  "dest-code": 9511,
  "quantity": 1344
}
```

Al termine, l'output dello stato `Map` è un array JSON, dove ogni item è l'output di un'iterazione. In questo caso, questo array contiene l'output della funzione `ship-val` Lambda.

Esempio di stato della mappa in linea con **`ItemSelector`**

Supponiamo che la funzione `ship-val` Lambda dell'esempio precedente richieda anche informazioni sul corriere della spedizione. Queste informazioni si aggiungono agli elementi dell'array per ogni iterazione. È possibile includere informazioni provenienti dall'input, insieme a informazioni specifiche sull'iterazione corrente dello `Map` stato. Notate il `ItemSelector` campo nell'esempio seguente:

```
"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
}
```

Il `ItemSelector` blocco sostituisce l'input delle iterazioni con un nodo JSON. Questo nodo contiene sia i dati dell'articolo corrente dall'oggetto contestuale sia le informazioni sul corriere dal campo dell'input dello `Map` stato. `delivery-partner` Di seguito è riportato un esempio di input per una singola iterazione. Lo `Map` stato passa questo input a una chiamata della funzione Lambda`ship-val`.

```
{
  "parcel": {
    "prod": "R31",
    "dest-code": 9511,
    "quantity": 1344
   },
   "courier": "UQS"
}
```

Nel precedente esempio di stato Inline Map, il `ResultPath` campo produce un output nello stesso formato dell'input. Tuttavia, sovrascrive il `detail.shipped` campo con un array in cui ogni elemento è l'output della chiamata Lambda `ship-val` di ogni iterazione.

Per ulteriori informazioni sull'utilizzo dello stato dello stato della mappa inline e dei relativi campi, consulta quanto segue.

- Ripeti un'azione utilizzando lo stato della mappa in linea
- Elaborazione di input e output in Step Functions
- ItemsPath
- Dati dell'oggetto contesto per gli stati Map

Elaborazione di input e **`Map`** output in stato di linea

Per un determinato `Map` stato, `InputPath`seleziona un sottoinsieme dell'input dello stato.

L'input di uno `Map` stato deve includere un array JSON. Lo `Map` stato esegue la `ItemProcessor` sezione una volta per ogni elemento dell'array. Se si specifica il `ItemsPath`campo, `Map` lo stato seleziona la posizione dell'input per trovare l'array su cui eseguire l'iterazione. Se non è specificato, il valore di `ItemsPath` è `$` e la sezione `ItemProcessor` prevede che l'array sia l'unico input. Se specificate il `ItemsPath` campo, il suo valore deve essere un percorso di riferimento. Lo `Map` stato applica questo percorso all'input effettivo dopo aver applicato il`InputPath`. `ItemsPath`Deve identificare un campo il cui valore è un array JSON.

L'input per ogni iterazione, per impostazione predefinita, è un singolo elemento del campo dell'array identificato dal `ItemsPath` valore. È possibile sovrascrivere questo valore con il `ItemSelector` campo.

Al termine, l'output dello stato `Map` è un array JSON, dove ogni item è l'output di un'iterazione.

Per ulteriori informazioni sugli input e gli output dello stato di Inline Map, consulta quanto segue:

- Ripeti un'azione utilizzando lo stato della mappa in linea
- Esempio di stato della mappa in linea con `ItemSelector`
- Elaborazione di input e output in Step Functions
- Dati dell'oggetto contesto per gli stati Map
- Elabora dinamicamente i dati con uno stato della mappa

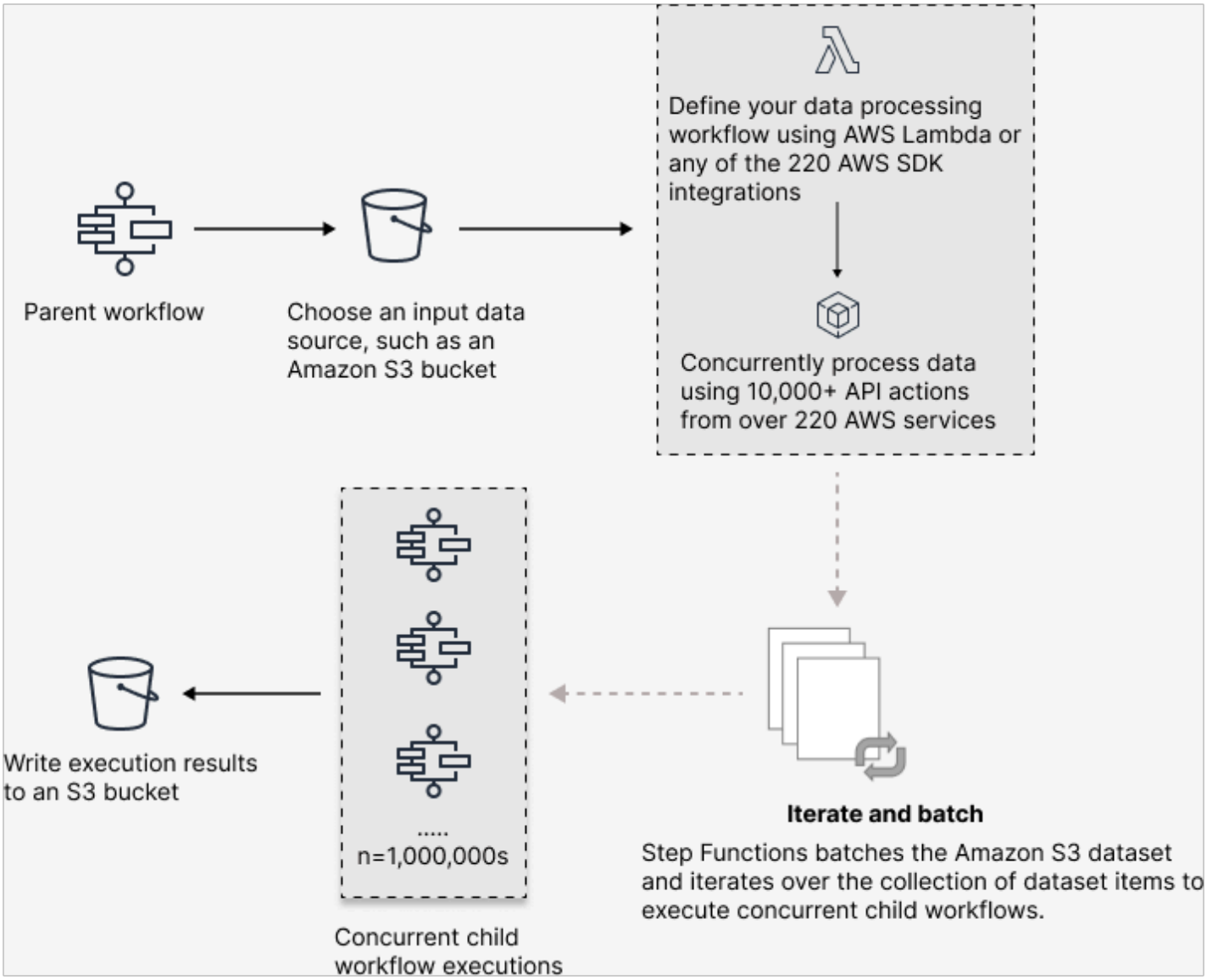
Utilizzo dello stato della mappa in modalità distribuita per orchestrare carichi di lavoro paralleli su larga scala

Con Step Functions, puoi orchestrare carichi di lavoro paralleli su larga scala per eseguire attività, come l'elaborazione su richiesta di dati semistrutturati. Questi carichi di lavoro paralleli consentono di elaborare contemporaneamente sorgenti di dati su larga scala archiviate in Amazon S3. Ad esempio, potresti elaborare un singolo file JSON o CSV che contiene grandi quantità di dati. Oppure potresti elaborare un ampio set di oggetti Amazon S3.

Per configurare un carico di lavoro parallelo su larga scala nei flussi di lavoro, includi uno `Map` stato in modalità Distribuita. Lo stato della mappa elabora gli elementi in un set di dati contemporaneamente. Uno `Map` stato impostato su Distributed è noto come stato Distributed Map. In modalità Distribuita, lo `Map` stato consente l'elaborazione ad alta concorrenza. In modalità distribuita, *`Map`* lo stato elabora gli elementi del set di dati in iterazioni denominate esecuzioni secondarie del flusso di lavoro. È possibile

specificare il numero di esecuzioni di workflow secondarie che possono essere eseguite in parallelo. Ogni esecuzione del flusso di lavoro secondario ha una propria cronologia di esecuzione separata da quella del flusso di lavoro principale. Se non lo specifichi, Step Functions esegue 10.000 esecuzioni parallele di flussi di lavoro secondari in parallelo.

La seguente illustrazione spiega come configurare carichi di lavoro paralleli su larga scala nei flussi di lavoro.



In questo argomento

- Termini chiave
- Esempio di definizione dello stato di Distributed Map
- Autorizzazioni per eseguire Distributed Map

- Campi relativi allo stato della mappa distribuita
- Passaggi successivi

Termini chiave

Modalità distribuita

Una modalità di elaborazione dello stato della mappa. In questa modalità, ogni iterazione dello `Map` stato viene eseguita come un'esecuzione secondaria del flusso di lavoro che consente un'elevata concorrenza. Ogni esecuzione del flusso di lavoro secondario ha la propria cronologia di esecuzione, che è separata dalla cronologia di esecuzione del flusso di lavoro principale. Questa modalità supporta la lettura di input da fonti di dati Amazon S3 su larga scala.

Stato della mappa distribuita

Uno stato della mappa impostato sulla modalità di elaborazione distribuita.

Workflow della mappa

Una serie di passaggi eseguiti da uno `Map` stato.

Flusso di lavoro principale

Un flusso di lavoro che contiene uno o più stati della mappa distribuita.

Esecuzione del workflow secondario

Un'iterazione dello stato della mappa distribuita. L'esecuzione di un flusso di lavoro secondario ha una propria cronologia di esecuzione, che è separata dalla cronologia di esecuzione del flusso di lavoro principale.

Esegui la mappa

Quando si esegue uno `Map` stato in modalità Distribuita, Step Functions crea una risorsa Map Run. Un Map Run si riferisce a un insieme di esecuzioni di workflow secondarie avviate da uno stato di Distributed Map e alle impostazioni di runtime che controllano queste esecuzioni. Step Functions assegna un Amazon Resource Name (ARN) a Map Run. È possibile esaminare un Map Run nella console Step Functions. Puoi anche richiamare l'azione `DescribeMapRun` API. Un Map Run emette anche metriche per. CloudWatch

Per ulteriori informazioni, consulta Esame di Map Run.

Esempio di definizione dello stato di Distributed Map

Usa lo `Map` stato in modalità Distribuita quando devi orchestrare carichi di lavoro paralleli su larga scala che soddisfano qualsiasi combinazione delle seguenti condizioni:

- La dimensione del set di dati supera i 256 KB.
- La cronologia degli eventi di esecuzione del flusso di lavoro supera le 25.000 voci.
- È necessaria una concorrenza di più di 40 iterazioni parallele.

Il seguente esempio di definizione dello stato di Distributed Map specifica il set di dati come file CSV archiviato in un bucket Amazon S3. Specifica inoltre una funzione Lambda che elabora i dati in ogni riga del file CSV. Poiché questo esempio utilizza un file CSV, specifica anche la posizione delle intestazioni delle colonne CSV. Per visualizzare la definizione completa di macchina a stati di questo esempio, consulta il tutorial Copiare dati CSV su larga scala utilizzando Distributed Map.

```
{
  "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"
      }
    }
  }
}
```

Autorizzazioni per eseguire Distributed Map

Quando includi uno stato della mappa distribuita nei flussi di lavoro, Step Functions necessita delle autorizzazioni appropriate per consentire al ruolo della macchina a stati di richiamare l'azione `StartExecution` API per lo stato della mappa distribuita.

Il seguente esempio di policy IAM concede i privilegi minimi richiesti al ruolo della macchina a stati per l'esecuzione dello stato della mappa distribuita.

> **Note**
>
> Assicurati di sostituirlo *stateMachineName* con il nome della macchina a stati in cui stai utilizzando lo stato Distributed Map. Ad esempio, `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:*"
    }
  ]
}
```

Inoltre, devi assicurarti di disporre dei privilegi minimi necessari per accedere alle AWS risorse utilizzate nello stato della mappa distribuita, come i bucket Amazon S3. Per informazioni, consulta Politiche IAM per l'utilizzo dello stato della mappa distribuita.

Campi relativi allo stato della mappa distribuita

Per utilizzare lo stato della mappa distribuita nei flussi di lavoro, specifica uno o più di questi campi. È possibile specificare questi campi in aggiunta ai campi di stato comuni.

`Type` (Obbligatorio)

Imposta il tipo di stato, ad esempio`Map`.

`ItemProcessor` (Obbligatorio)

Contiene i seguenti oggetti JSON che specificano la modalità e la definizione di elaborazione `Map` dello stato.

- `ProcessorConfig`— Un oggetto JSON che specifica la configurazione per lo stato. `Map` Questo oggetto contiene i seguenti sottocampi:
  - `Mode`— Impostato per **`DISTRIBUTED`** utilizzare lo `Map` stato in modalità Distribuita.

> **Note**
>
> Attualmente, se si utilizza `Map` lo stato all'interno dei flussi di lavoro Express, non è possibile `Mode` impostarlo `DISTRIBUTED` su. Tuttavia, se utilizzi `Map` lo stato all'interno dei flussi di lavoro Standard, puoi impostarlo su. `Mode` `DISTRIBUTED`

  - `ExecutionType`— specifica il tipo di esecuzione per il flusso di lavoro Map come STANDARD o EXPRESS. È necessario fornire questo campo se è stato specificato `DISTRIBUTED` per il `Mode` sottocampo. Per ulteriori informazioni sui tipi di flusso di lavoro, vedereFlussi di lavoro Standard e Express.
- `StartAt`— specifica una stringa che indica il primo stato di un flusso di lavoro. Questa stringa fa distinzione tra maiuscole e minuscole e deve corrispondere al nome di uno degli oggetti di stato. Questo stato viene eseguito per primo per ogni elemento del set di dati. Qualsiasi input di esecuzione fornito allo `Map` stato passa per primo allo `StartAt` stato.
- `States`— Un oggetto JSON contenente un set di stati delimitato da virgole. In questo oggetto, si definisce il. Map workflow

`ItemReader`

Speciifica un set di dati e la sua posizione. Lo `Map` stato riceve i dati di input dal set di dati specificato.

In modalità distribuita, puoi utilizzare un payload JSON passato da uno stato precedente o un'origine dati Amazon S3 su larga scala come set di dati. Per ulteriori informazioni, consulta ItemReader.

`ItemsPath` (facoltativo).

Speciifica un percorso di riferimento utilizzando la JsonPathsintassi per selezionare il nodo JSON che contiene una matrice di elementi all'interno dell'input dello stato.

In modalità distribuita, si specifica questo campo solo quando si utilizza un array JSON di un passaggio precedente come input di stato. Per ulteriori informazioni, consulta ItemsPath.

`ItemSelector` (facoltativo).

Sostituisce i valori dei singoli elementi del set di dati prima che vengano passati a ciascuna iterazione di stato. `Map`

In questo campo, specifichi un input JSON valido che contiene una raccolta di coppie chiave-valore. Queste coppie possono essere valori statici definiti nella definizione della macchina a stati,

valori selezionati dall'input di stato utilizzando un percorso o valori a cui si accede dall'oggetto di contesto. Per ulteriori informazioni, consulta ItemSelector.

`ItemBatcher` (facoltativo).

Speciifica di elaborare gli elementi del set di dati in batch. Ogni esecuzione secondaria del flusso di lavoro riceve quindi un batch di questi elementi come input. Per ulteriori informazioni, consulta ItemBatcher.

`MaxConcurrency` (facoltativo).

Speciifica il numero di esecuzioni di workflow secondarie che possono essere eseguite in parallelo. L'interprete consente solo fino al numero specificato di esecuzioni parallele di flussi di lavoro secondari. Se non specificate un valore di concorrenza o lo impostate su zero, Step Functions non limita la concorrenza ed esegue 10.000 esecuzioni parallele di workflow secondari.

> **Note**
>
> Sebbene sia possibile specificare un limite di concorrenza più elevato per le esecuzioni di flussi di lavoro secondari paralleli, si consiglia di non superare la capacità di un AWS servizio downstream, ad esempio. AWS Lambda

`MaxConcurrencyPath` (facoltativo).

Se desideri fornire un valore di concorrenza massimo in modo dinamico dall'input dello stato utilizzando un percorso di riferimento, usa. `MaxConcurrencyPath` Una volta risolto, il percorso di riferimento deve selezionare un campo il cui valore è un numero intero non negativo.

> **Note**
>
> Uno `Map` stato non può includere entrambi `MaxConcurrency` e. `MaxConcurrencyPath`

`ToleratedFailurePercentage` (facoltativo).

Definisce la percentuale di elementi non riusciti da tollerare in una Map Run. La Map Run fallisce automaticamente se supera questa percentuale. Step Functions calcola la percentuale di articoli non riusciti come risultato del numero totale di articoli non riusciti o scaduti diviso per il numero totale di articoli. È necessario specificare un valore compreso tra zero e 100. Per ulteriori informazioni, consulta Soglia di errore tollerata per lo stato della mappa distribuita.

`ToleratedFailurePercentagePath` (facoltativo).

Se si desidera fornire dinamicamente un valore percentuale di errore tollerato in base allo stato immesso utilizzando un percorso di riferimento, utilizzare. `ToleratedFailurePercentagePath` Una volta risolto, il percorso di riferimento deve selezionare un campo il cui valore è compreso tra zero e 100.

`ToleratedFailureCount` (facoltativo).

Definisce il numero di elementi non riusciti da tollerare in una Map Run. La Map Run fallisce automaticamente se supera questo numero. Per ulteriori informazioni, consulta Soglia di errore tollerata per lo stato della mappa distribuita.

`ToleratedFailureCountPath` (facoltativo).

Se si desidera fornire dinamicamente un valore di conteggio degli errori tollerati in base allo stato immesso utilizzando un percorso di riferimento, utilizzare. `ToleratedFailureCountPath` Una volta risolto, il percorso di riferimento deve selezionare un campo il cui valore è un numero intero non negativo.

`Label` (facoltativo).

Una stringa che identifica in modo univoco uno stato. `Map` Per ogni Map Run, Step Functions aggiunge l'etichetta al Map Run ARN. Di seguito è riportato un esempio di Map Run ARN con un'etichetta personalizzata denominata: `demoLabel`

```
arn:aws:states:us-east-1:123456789012:mapRun:demoWorkflow/
demoLabel:3c39a231-69bb-3d89-8607-9e124eddbb0b
```

Se non specificate un'etichetta, Step Functions genera automaticamente un'etichetta univoca.

> **Note**
>
> Le etichette non possono superare i 40 caratteri di lunghezza, devono essere univoche all'interno di una definizione di macchina a stati e non possono contenere nessuno dei seguenti caratteri:
>
> - Caratteri di spazio bianco
> - Caratteri jolly () ? *
> - Caratteri tra parentesi quadre () < > { } [ ]
> - Caratteri speciali () : ; , \ | ^ ~ $ # % & ` "

- Caratteri di controllo (\\u0000- \\u001f o \\u007f -\\u009f).

Step Functions consente di creare nomi per macchine a stati, esecuzioni, attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

`ResultWriter` (facoltativo).

Speciifica la posizione Amazon S3 in cui Step Functions scrive tutti i risultati di esecuzione dei flussi di lavoro secondari.

Step Functions consolida tutti i dati di esecuzione del workflow secondario, come input e output dell'esecuzione, ARN e stato di esecuzione. Quindi esporta le esecuzioni con lo stesso stato nei rispettivi file nella posizione Amazon S3 specificata. Per ulteriori informazioni, consulta ResultWriter.

Se non esporti i risultati dello `Map` stato, restituisce una matrice di tutti i risultati di esecuzione del flusso di lavoro secondario. Per esempio:

```
[1, 2, 3, 4, 5]
```

`ResultPath` (facoltativo).

Speciifica dove, nell'input, inserire l'output delle iterazioni. L'input viene quindi filtrato come specificato dal `OutputPath`campo, se presente, prima di essere passato come output dello stato. Per ulteriori informazioni, consultare Elaborazione di input e output.

`ResultSelector` (facoltativo).

Passa una raccolta di coppie chiave-valore, in cui i valori sono statici o selezionati dal risultato. Per ulteriori informazioni, consulta ResultSelector.

> **Tip**
>
> Se lo stato Parallel o Map che usi nelle tue macchine a stati restituisce un array di array, puoi trasformarli in un array flat con il campo. ResultSelector Per ulteriori informazioni, consulta Appiattimento di una serie di array.

`Retry` (facoltativo).

Una serie di oggetti, denominati Retriers, che definiscono una politica di nuovi tentativi. Un'esecuzione utilizza la politica di riprova se lo stato rileva errori di runtime. Per ulteriori informazioni, consulta Esempi di macchine a stati che utilizzano Retry e Catch.

> **Note**
>
> Se si definiscono Retriers per lo stato Distributed Map, la politica di nuovo tentativo si applica a tutte le esecuzioni di workflow secondarie avviate dallo stato. `Map` Ad esempio, immaginate che il vostro `Map` stato abbia avviato tre esecuzioni secondarie di workflow, di cui una fallisce. Quando si verifica l'errore, l'esecuzione utilizza il `Retry` campo, se definito, per lo `Map` stato. La politica dei nuovi tentativi si applica a tutte le esecuzioni dei flussi di lavoro secondari e non solo a quelle non riuscite. Se una o più esecuzioni di workflow secondarie falliscono, Map Run non riesce.
> Quando si riprova uno `Map` stato, viene creata una nuova Map Run.

`Catch` (facoltativo).

Un array di oggetti, denominati catcher, che definiscono uno stato di fallback. Step Functions utilizza i Catcher definiti in `Catch` se lo stato rileva errori di runtime. Quando si verifica un errore, l'esecuzione utilizza innanzitutto tutti i retrier definiti in. `Retry` Se la politica di riprova non è definita o è esaurita, l'esecuzione utilizza i relativi Catcher, se definiti. Per ulteriori informazioni, consulta Stati di fallback.

Passaggi successivi

Per continuare a saperne di più sullo stato della mappa distribuita, consulta le seguenti risorse:

- Elaborazione di input e output

  Per configurare l'input ricevuto da uno stato della Mappa Distribuita e l'output che genera, Step Functions fornisce i seguenti campi:

  - ItemReader
  - ItemsPath
  - ItemSelector
  - ItemBatcher

- ResultWriter
- Analisi di un file CSV di input

Oltre a questi campi, Step Functions offre anche la possibilità di definire una soglia di errore tollerata per Distributed Map. Questo valore consente di specificare il numero massimo o la percentuale di elementi non riusciti come soglia di errore per una Map Run. Per ulteriori informazioni sulla configurazione della soglia di errore tollerata, vedere. Soglia di errore tollerata per lo stato della mappa distribuita

- Utilizzo dello stato della mappa distribuita

  Fai riferimento ai seguenti tutorial e progetti di esempio per iniziare a utilizzare Distributed Map state.

  - Guida introduttiva all'utilizzo dello stato di Distributed Map
  - Elaborazione di un intero batch di dati con una funzione Lambda
  - Elaborazione di singoli elementi di dati con una funzione Lambda
  - Progetto di esempio: elaborazione di un file CSV con Distributed Map
  - Progetto di esempio: elabora i dati in un bucket Amazon S3 con Distributed Map
- Esamina l'esecuzione dello stato della mappa distribuita

  La console Step Functions fornisce una pagina Map Run Details, che mostra tutte le informazioni relative all'esecuzione dello stato di Distributed Map. Per informazioni su come esaminare le informazioni visualizzate in questa pagina, vedereEsame di Map Run.

Soglia di errore tollerata per lo stato della mappa distribuita

Quando orchestri carichi di lavoro paralleli su larga scala, puoi anche definire una soglia di errore tollerata. Questo valore consente di specificare il numero massimo o la percentuale di elementi non riusciti come soglia di errore per una Map Run. A seconda del valore specificato, Map Run fallisce automaticamente se supera la soglia. Se si specificano entrambi i valori, il flusso di lavoro fallisce quando supera uno dei due valori.

Specificare una soglia consente di fallire un numero specifico di elementi prima che l'intero Map Run abbia esito negativo. Step Functions restituisce un `States.ExceedToleratedFailureThreshold` errore quando Map Run fallisce perché viene superata la soglia specificata.

> **Note**
>
> Step Functions può continuare a eseguire flussi di lavoro secondari in una Map Run anche dopo il superamento della soglia di errore tollerata, ma prima che Map Run fallisca.

Per specificare il valore di soglia in Workflow Studio, selezionare Imposta una soglia di errore tollerata in Configurazione aggiuntiva nel campo Impostazioni di runtime.

Percentuale di fallimento tollerato

Definisce la percentuale di articoli non riusciti da tollerare. Il Map Run fallisce se questo valore viene superato. Step Functions calcola la percentuale di articoli non riusciti come risultato del numero totale di articoli non riusciti o scaduti diviso per il numero totale di articoli. È necessario specificare un valore compreso tra zero e 100. Il valore percentuale predefinito è zero, il che significa che il flusso di lavoro ha esito negativo se una delle sue esecuzioni secondarie del flusso di lavoro fallisce o scade. Se si specifica la percentuale come 100, il flusso di lavoro non avrà esito negativo anche se tutte le esecuzioni del flusso di lavoro secondario falliscono.

In alternativa, è possibile specificare la percentuale come percorso di riferimento per una coppia chiave-valore esistente nell'input di stato della Mappa distribuita. Questo percorso deve risolversi in un numero intero positivo compreso tra 0 e 100 in fase di esecuzione. Il percorso di riferimento viene specificato nel `ToleratedFailurePercentagePath` sottocampo.

Ad esempio, con il seguente input:

```
{
  "percentage": 15
}
```

È possibile specificare la percentuale utilizzando un percorso di riferimento a tale input come segue:

```
{
  ...
  "Map": {
    "Type": "Map",
    ...
    "ToleratedFailurePercentagePath": "$.percentage"
```

```
    ...
  }
}
```

> **Important**
>
> È possibile specificare uno `ToleratedFailurePercentage` o entrambi`ToleratedFailurePercentagePath`, ma non entrambi nella definizione dello stato della Mappa Distribuita.

Numero di fallimenti tollerati

Definisce il numero di elementi non riusciti da tollerare. Il Map Run fallisce se questo valore viene superato.

In alternativa, puoi specificare il conteggio come percorso di riferimento per una coppia chiave-valore esistente nell'input di stato della Mappa distribuita. Questo percorso deve risolversi in un numero intero positivo in fase di esecuzione. Il percorso di riferimento viene specificato nel `ToleratedFailureCountPath` sottocampo.

Ad esempio, con il seguente input:

```
{
  "count": 10
}
```

È possibile specificare il numero utilizzando un percorso di riferimento a tale input come segue:

```
{
  ...
  "Map": {
    "Type": "Map",
    ...
    "ToleratedFailureCountPath": "$.count"
    ...
  }
}
```

> **Important**
>
> È possibile specificare uno `ToleratedFailureCount` o entrambi`ToleratedFailureCountPath`, ma non entrambi nella definizione dello stato della Mappa Distribuita.

Transizioni

Quando si avvia una nuova esecuzione della macchina a stati, il sistema inizia con lo stato a cui si fa riferimento nel campo di primo livello`StartAt`. Questo campo, fornito come stringa, deve corrispondere esattamente, maiuscole e minuscole, al nome di uno stato nel flusso di lavoro.

Dopo l'esecuzione di uno stato, AWS Step Functions utilizza il valore del `Next` campo per determinare lo stato successivo a cui passare.

`Next`i campi specificano anche i nomi degli stati sotto forma di stringhe. Questa stringa fa distinzione tra maiuscole e minuscole e deve corrispondere esattamente al nome di uno stato specificato nella descrizione della macchina a stati

Lo stato seguente, ad esempio, include una transizione a `NextState`.

```
"SomeState" : {
  ...,
  "Next" : "NextState"
}
```

La maggior parte degli stati consente una sola regola di transizione con il `Next` campo. Tuttavia, alcuni stati di controllo del flusso, come uno `Choice` stato, consentono di specificare più regole di transizione, ognuna con il proprio campo. `Next` Amazon States Language fornisce dettagli su ogni tipo di stato che è possibile specificare, con informazioni su come specificare le transizioni.

Gli stati possono contenere più transizioni in ingresso da altri stati.

Il processo si ripete finché non raggiunge uno stato terminale (uno stato con `"Type": Succeed"Type": Fail`, o`"End": true`) o si verifica un errore di runtime.

Quando si redriveesegue un'esecuzione, viene considerata una transizione di stato. Inoltre, tutti gli stati che vengono eseguiti nuovamente in a redrive sono considerati transizioni di stato.

Le regole seguenti si applicano agli stati in una macchina a stati:

- Gli stati possono verificarsi in qualsiasi ordine all'interno del blocco che lo racchiude. Tuttavia, l'ordine in cui sono elencati non influisce sull'ordine in cui vengono eseguiti. che dipende dal contenuto degli stati stessi.
- All'interno di una macchina a stati, può esserci un solo stato designato come `start` stato. Lo `start` stato è definito dal valore del `StartAt` campo nella struttura di primo livello.
- A seconda della logica della macchina a stati, ad esempio, se la macchina a stati ha più rami logici, è possibile avere più di uno `end` stato.
- Se la macchina a stati è composta da un solo stato, può essere sia lo stato iniziale che quello finale.

Transizioni nello stato della mappa distribuita

Quando utilizzi lo `Map` stato in modalità Distribuita, ti verrà addebitata una transizione di stato per ogni esecuzione del workflow secondario avviata dallo stato della Mappa Distribuita. Quando utilizzi lo `Map` stato in modalità Inline, non ti viene addebitata una transizione di stato per ogni iterazione dello stato della Mappa in linea.

È possibile ottimizzare i costi utilizzando lo `Map` stato in modalità Distribuita e includere un flusso di lavoro annidato nella definizione dello stato. `Map` Lo stato Distributed Map aggiunge inoltre ulteriore valore quando si avviano esecuzioni secondarie di workflow di tipo Express. Step Functions memorizza la risposta e lo stato delle esecuzioni dei flussi di lavoro secondari Express, il che riduce la necessità di archiviare i dati di esecuzione nei CloudWatch log. Puoi anche accedere ai controlli di flusso disponibili con uno stato Distributed Map, come la definizione di soglie di errore o il raggruppamento di un gruppo di elementi. Per informazioni sui prezzi di Step Functions, consulta la pagina AWS Step Functionsdei prezzi.

Dati delle macchine a stati

I dati delle macchine a stati assumono le forme seguenti:

- Input iniziale in una macchina a stati
- Dati passati tra gli stati
- Output di una macchina a stati

Questa sezione illustra il formato e l'utilizzo dei dati delle macchine a stati in AWS Step Functions.

Argomenti

- Formato dei dati
- Input/output delle macchine a stati
- Input/output degli stati

Formato dei dati

I dati della macchina statale sono rappresentati da testo JSON. Puoi fornire valori a una macchina a stati utilizzando qualsiasi tipo di dati supportato da JSON.

> **Note**
>
> - I numeri in formato testo JSON sono conformi alla semantica. JavaScript Tali numeri corrispondono in genere a valori IEEE-854 a precisione doppia.
> - Quanto segue è un testo JSON valido:
>   - Stringhe autonome, delimitate da virgolette
>   - Oggetti
>   - Matrici
>   - Numeri
>   - Valori booleani
>   - `null`
> - L'output di uno stato diventa l'input per lo stato successivo. Tuttavia, è possibile limitare gli stati a lavorare su un sottoinsieme dei dati di input utilizzando Input and Output Processing.

Input/output delle macchine a stati

È possibile fornire i dati di input iniziali a una macchina a AWS Step Functions stati in due modi. È possibile passare i dati a un'`StartExecution`azione quando si avvia un'esecuzione. È inoltre possibile passare i dati alla state machine dalla console Step Functions. I dati iniziali vengono passati allo stato `StartAt` della macchina a stati. Se viene fornito alcun input, per default viene utilizzato un oggetto vuoto (`{}`).

L'output dell'esecuzione viene restituito dall'ultimo stato (`terminal`). Tale output viene riportato come testo JSON nel risultato dell'esecuzione.

Per i flussi di lavoro standard, puoi recuperare i risultati dell'esecuzione dalla cronologia delle esecuzioni utilizzando chiamanti esterni, come l'azione. `DescribeExecution` È possibile visualizzare i risultati dell'esecuzione nella console Step Functions.

Per Express Workflow, se hai abilitato la registrazione, puoi recuperare i risultati dai CloudWatch log o visualizzare ed eseguire il debug delle esecuzioni nella console Step Functions. Per ulteriori informazioni, consultare Registrazione utilizzandoCloudWatchRegistri e Visualizzazione e debug delle esecuzioni sulla console Step Functions.

Dovresti anche considerare le quote relative alla tua macchina statale. Per ulteriori informazioni, consulta Quote.

Input/output degli stati

L'input di ogni stato è costituito da testo JSON proveniente dallo stato precedente o, per lo stato `StartAt`, dall'input per l'esecuzione. Alcuni stati del controllo di flusso ripetono l'input nell'output.

Nell'esempio seguente, la macchina a stati aggiunge due numeri insieme.

1. Definire la funzione 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. Definisci la macchina a stati .

```
{
  "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. Avviare un'esecuzione con il testo JSON seguente.

```
{ "numbers": [3, 4] }
```

Lo Add stato riceve il testo JSON e lo passa alla funzione Lambda.

La funzione Lambda restituisce il risultato del calcolo allo stato.

Lo stato restituisce il valore seguente nell'output.

```
{ "result": 7 }
```

Poiché Add è anche lo stato finale nella macchina a stati, questo valore viene restituito come output della macchina a stati.

Se lo stato finale non restituisce alcun output, la macchina restituisce un oggetto vuoto ({}).

Per ulteriori informazioni, consulta Elaborazione di input e output in Step Functions.

Elaborazione di input e output in Step Functions

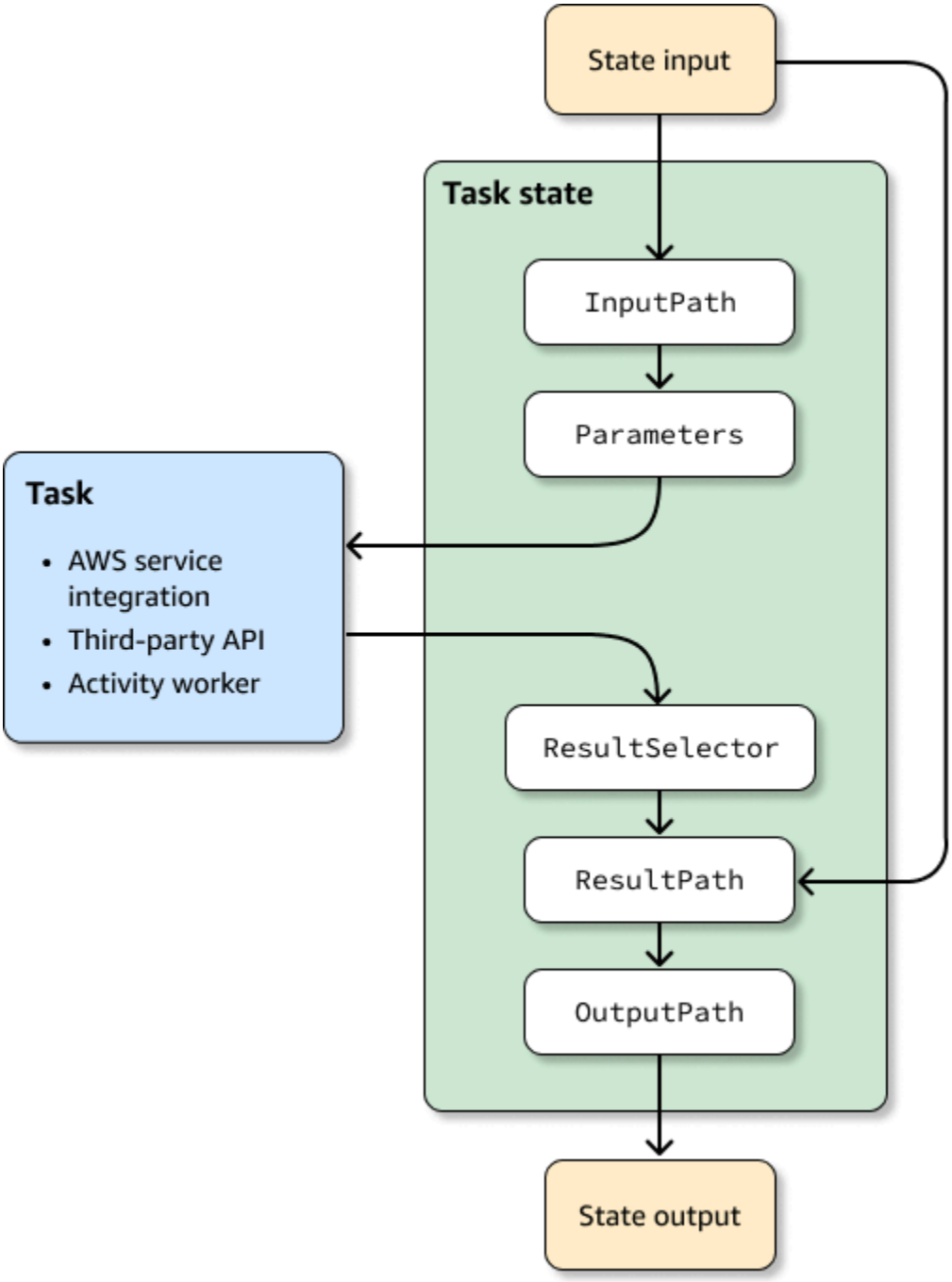
Un'esecuzione di Step Functions riceve un testo JSON come input e lo passa al primo stato del flusso di lavoro. I singoli stati ricevono JSON come input e in genere trasferiscono JSON come output allo stato successivo. Comprendere il modo in cui questo flusso di informazioni passa da uno stato all'altro e imparare a filtrare e manipolare questi dati è fondamentale per progettare e implementare flussi di lavoro in AWS Step Functions in maniera efficace.

In Amazon States Language, questi campi filtrano e controllano il flusso di JSON da uno stato all'altro:

- `InputPath`
- `Parameters`

- `ResultSelector`
- `ResultPath`
- `OutputPath`

Il diagramma seguente mostra come le informazioni JSON passano attraverso uno stato di attività. `InputPath`seleziona quali parti dell'input JSON passare all'attività dello `Task` stato (ad esempio, una funzione). AWS Lambda `ResultPath`seleziona quindi la combinazione dell'input dello stato e del risultato dell'attività da passare all'output. `OutputPath`può filtrare l'output JSON per limitare ulteriormente le informazioni che vengono passate all'output.



`InputPath`,`Parameters`, `ResultSelectorResultPath`, e `OutputPath` ciascuno manipola JSON mentre si muove attraverso ogni stato del flusso di lavoro.

Ciascuno può utilizzare percorsi per selezionare parti del JSON dall'input o dal risultato. Un percorso è una stringa, che inizia con$, che identifica i nodi all'interno del testo JSON. I percorsi Step Functions utilizzano la JsonPathsintassi.

> **Tip**
>
> Usa il simulatore di flusso di dati nella console Step Functions per testare la sintassi del percorso JSON, comprendere meglio come i dati vengono manipolati all'interno di uno stato e vedere come i dati vengono trasferiti tra gli stati.

> **Tip**
>
> Per implementare un esempio di flusso di lavoro che include l'elaborazione di input e output nel vostro computer Account AWS, consultate il Modulo 6 - Elaborazione di input e output di The Workshop. AWS Step Functions

Argomenti

- Percorsi
- InputPath, Parametri e ResultSelector
- ResultPath
- OutputPath
- InputPathResultPath, ed OutputPath esempi
- Campi di input e output dello stato della mappa
- Oggetto Context

Percorsi

In Amazon States Language, un percorso è una stringa che inizia con e $ che puoi utilizzare per identificare i componenti all'interno del testo JSON. I percorsi seguono la JsonPathsintassi. È possibile specificare un percorso per accedere a sottoinsiemi di input quando si specificano i valori di `InputPath`, `ResultPath` e `OutputPath`. Per ulteriori informazioni, consulta Elaborazione di input e output in Step Functions.

> **Note**
>
> È anche possibile specificare un nodo JSON di input o un oggetto contesto utilizzando percorsi all'interno del campo `Parameters` dello stato di una definizione. Per informazioni, consulta Passa i parametri a un'API di servizio.

È necessario utilizzare la notazione tra parentesi quadre se il nome del campo contiene caratteri non inclusi nella `member-name-shorthand` definizione della JsonPath regola ABNF. Pertanto, per codificare caratteri speciali, come i segni di punteggiatura (esclusi_), è necessario utilizzare la notazione tra parentesi quadre. Ad esempio, `$.abc.['def ghi']`.

Percorsi di riferimento

Un percorso di riferimento è un percorso la cui sintassi è limitata in modo tale che può identificare solo un singolo nodo in una struttura JSON:

- Puoi accedere a campi di oggetto unicamente con la notazione del punto (`.`) e tra parentesi quadra (`[ ]`).
- Le funzioni come `length()` non sono supportate.
- Operatori lessicali, che non sono simbolici, ad esempio non sono supportati. `subsetof`
- Il filtraggio tramite espressioni regolari o facendo riferimento a un altro valore nella struttura JSON non è supportato.
- L'`@`operatore, che corrisponde al nodo corrente in fase di elaborazione in un filtro, non corrisponde ai valori scalari. Corrisponde solo agli oggetti.

Ad esempio, se i dati di input dello stato contengono i seguenti valori:

```
{
  "foo": 123,
  "bar": ["a", "b", "c"],
  "car": {
      "cdr": true
  },
  "jar": [{"a": 1}, {"a": 5}, {"a": 2}, {"a": 7}, {"a": 3}]
}
```

I percorsi di riferimento seguenti restituirebbero quanto segue.

```
$.foo => 123
$.bar => ["a", "b", "c"]
$.car.cdr => true
$.jar[?(@.a >= 5)] => [{"a": 5}, {"a": 7}]
```

Alcuni stati utilizzano percorsi e percorsi di riferimento per controllare il flusso di una macchina a stati o configurare impostazioni o opzioni di uno stato. Per ulteriori informazioni, consulta Modellazione dell'elaborazione del percorso di input e output del flusso di lavoro con il simulatore di flusso di dati e Utilizzo efficace di JsonPath in. AWS Step Functions

Appiattimento di una serie di array

Se lo Eseguire la mappatura stato Parallel o nelle tue macchine a stati restituisce un array di array, puoi trasformarli in una matrice piatta con il campo. ResultSelector Puoi includere questo campo nella definizione dello stato Parallel o Map per manipolare il risultato di questi stati.

Per appiattire gli array, utilizzate la sintassi JMESPath nel `ResultSelector` campo, come illustrato `[*]` nell'esempio seguente.

```
"ResultSelector": {
    "flattenArray.$": "$[*][*]"
  }
```

Per esempi che mostrano come appiattire un array, consultate il passaggio 3 nei seguenti tutorial:

- Elaborazione di un intero batch di dati con una funzione Lambda
- Elaborazione di singoli elementi di dati con una funzione Lambda

InputPath, Parametri e ResultSelector

I `InputPath` `ResultSelector` campi `Parameters` e forniscono un modo per manipolare JSON durante il flusso di lavoro. `InputPath`può limitare l'input passato filtrando la notazione JSON utilizzando un percorso (vedi). Percorsi Il campo `Parameters` consente di inoltrare una raccolta di coppie chiave-valore, dove i valori sono valori statici definiti nella definizione della macchina a stati o selezionati dall'input utilizzando un percorso. Il `ResultSelector` campo fornisce un modo per manipolare il risultato dello stato prima che venga applicato. `ResultPath`

AWS Step Functions applica prima il `InputPath` campo e poi il `Parameters` campo. Innanzitutto puoi filtrare l'input grezzo in modo da avere la selezione che desideri utilizzando `InputPath` e

quindi applicare `Parameters` per manipolare ulteriormente l'input o aggiungere nuovi valori. È quindi possibile utilizzare il `ResultSelector` campo per manipolare l'output dello stato prima che `ResultPath` venga applicato.

> **Tip**
>
> Usa il simulatore di flusso di dati nella console Step Functions per testare la sintassi del percorso JSON, comprendere meglio come i dati vengono manipolati all'interno di uno stato e vedere come i dati vengono trasferiti tra gli stati.

InputPath

Utilizza `InputPath` per selezionare una parte dell'input dello stato.

Se, ad esempio, l'input per il tuo stato include quanto segue:

```
{
  "comment": "Example for InputPath.",
  "dataset1": {
    "val1": 1,
    "val2": 2,
    "val3": 3
  },
  "dataset2": {
    "val1": "a",
    "val2": "b",
    "val3": "c"
  }
}
```

Puoi applicare l'`InputPath`.

```
"InputPath": "$.dataset2",
```

Con il precedente `InputPath`, il JSON trasmesso come input è il seguente.

```
{
  "val1": "a",
```

```
  "val2": "b",
  "val3": "c"
}
```

> **Note**
>
> Un percorso può produrre una selezione di valori. Analizza l'esempio seguente.
>
> ```
> { "a": [1, 2, 3, 4] }
> ```
>
> Se applichi il percorso `$.a[0:2]`, il risultato è il seguente.
>
> ```
> [ 1, 2 ]
> ```

Parametri

Questa sezione descrive i diversi modi in cui è possibile utilizzare il campo Parametri.

Coppie chiave/valore

Utilizzate il `Parameters` campo per creare una raccolta di coppie chiave-valore che vengono passate come input. I singoli valori possono essere o valori statici inclusi nella definizione della tua macchina a stati o valori selezionati dall'input o dall'oggetto contestuale mediante un percorso. Nel caso delle coppie chiave-valore in cui il valore viene selezionato mediante un percorso, il nome della chiave deve terminare con `.$`.

Ad esempio, supponiamo di fornire il seguente input.

```
{
  "comment": "Example for Parameters.",
  "product": {
    "details": {
       "color": "blue",
       "size": "small",
       "material": "cotton"
    },
    "availability": "in stock",
    "sku": "2317",
    "cost": "$23"
```

```
  }
}
```

Per selezionare alcune informazioni puoi specificare questi parametri nella definizione della tua macchina a stati.

```
"Parameters": {
        "comment": "Selecting what I care about.",
        "MyDetails": {
          "size.$": "$.product.details.size",
          "exists.$": "$.product.availability",
          "StaticValue": "foo"
        }
      },
```

A partire dall'input e dal campo `Parameters` precedenti, questo è l'oggetto JSON che viene trasmesso.

```
{
  "comment": "Selecting what I care about.",
  "MyDetails": {
      "size": "small",
      "exists": "in stock",
      "StaticValue": "foo"
  }
},
```

Oltre all'input, è possibile accedere a un oggetto JSON speciale, noto come oggetto contestuale. L'oggetto contestuale include informazioni sull'esecuzione della macchina a stati. Per informazioni, consulta Oggetto Context.

Risorse connesse

Il campo `Parameters` è inoltre in grado di trasmettere informazioni alle risorse connesse. Ad esempio, se lo stato dell'attività è l'orchestrazione di un AWS Batch lavoro, è possibile passare i parametri API pertinenti direttamente alle azioni API di quel servizio. Per ulteriori informazioni, consultare:

- Passa i parametri a un'API di servizio
- Uso di altri servizi

Amazon S3

Se i dati della funzione Lambda che trasmetti tra gli stati potrebbero superare i 262.144 byte, ti consigliamo di utilizzare Amazon S3 per archiviare i dati e implementare uno dei seguenti metodi:

- Utilizza lo stato Distributed Map nel tuo flusso di lavoro in modo che lo `Map` stato possa leggere l'input direttamente dalle fonti di dati Amazon S3. Per ulteriori informazioni, consulta Utilizzo dello stato della mappa in modalità distribuita.
- Analizza l'Amazon Resource Name (ARN) del bucket nel parametro per ottenere il nome `Payload` del bucket e il valore della chiave. Per ulteriori informazioni, consulta Usa gli ARN Amazon S3 invece di trasferire payload di grandi dimensioni.

In alternativa, puoi modificare l'implementazione per trasferire payload più piccoli nelle tue esecuzioni.

ResultSelector

Utilizzate il `ResultSelector` campo per manipolare il risultato di uno stato prima `ResultPath` che venga applicato. Il `ResultSelector` campo consente di creare una raccolta di coppie chiave-valore, in cui i valori sono statici o selezionati dal risultato dello stato. Utilizzando il `ResultSelector` campo, puoi scegliere quali parti del risultato di uno stato vuoi passare al `ResultPath` campo.

> **Note**
>
> Con il `ResultPath` campo, puoi aggiungere l'output del `ResultSelector` campo all'input originale.

`ResultSelector`è un campo opzionale nei seguenti stati:

- Eseguire la mappatura
- Attività
- Parallel

Ad esempio, le integrazioni del servizio Step Functions restituiscono metadati oltre al payload nel risultato. `ResultSelector`può selezionare parti del risultato e unirle allo stato di input con. `ResultPath` In questo esempio, vogliamo selezionare solo la `resourceType` e e `ClusterId` unirla con l'input di stato di un Amazon EMR CreateCluster.sync. Considerato quanto segue:

```
{
  "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"
  }
}
```

È quindi possibile selezionare `resourceType` e `ClusterId` utilizzare`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"
}
```

Con l'input dato, l'utilizzo `ResultSelector` produce:

```
{
  "OtherDataFromInput": {},
  "EMROutput": {
    "ResourceType": "elasticmapreduce",
```

```
    "ClusterId": "AKIAIOSFODNN7EXAMPLE"
  }
}
```

Appiattimento di una serie di array

Se lo Eseguire la mappatura stato Parallel o nelle tue macchine a stati restituisce un array di array, puoi trasformarli in una matrice piatta con il campo. ResultSelector Puoi includere questo campo nella definizione dello stato Parallel o Map per manipolare il risultato di questi stati.

Per appiattire gli array, utilizzate la sintassi JMESPath nel `ResultSelector` campo, come illustrato `[*]` nell'esempio seguente.

```
"ResultSelector": {
    "flattenArray.$": "$[*][*]"
  }
```

Per esempi che mostrano come appiattire un array, consultate il passaggio 3 nei seguenti tutorial:

- Elaborazione di un intero batch di dati con una funzione Lambda
- Elaborazione di singoli elementi di dati con una funzione Lambda

ResultPath

L'output di uno stato può essere una copia dell'input, il risultato generato (ad esempio, l'output ottenuto da una funzione Lambda di uno stato `Task`) o una combinazione dell'input e del risultato. Utilizza `ResultPath` per controllare quale di queste combinazioni verrà passata all'output dello stato.

I tipi di stato seguenti possono generare un risultato e includere `ResultPath:`

- Pass
- Attività
- Parallel
- Eseguire la mappatura

Utilizza `ResultPath` per combinare il risultato di un task con l'input del task o selezionare una di queste opzioni. Il percorso specificato in `ResultPath` controlla le informazioni passate all'output.

> **Note**
>
> `ResultPath` è limitato all'utilizzo di percorsi di riferimento, che limitano l'ambito consentendo l'identificazione di un unico nodo in JSON. Consultare Percorsi di riferimento nella sezione Amazon States Language.

Questi esempi si basano sulla macchina a stati e sulla funzione Lambda descritte nel Creazione di una macchina a stati Step Functions che utilizza Lambda tutorial. Consultare il tutorial e testa diversi output provando vari percorsi in un campo `ResultPath`.

È possibile utilizzare ResultPath per:

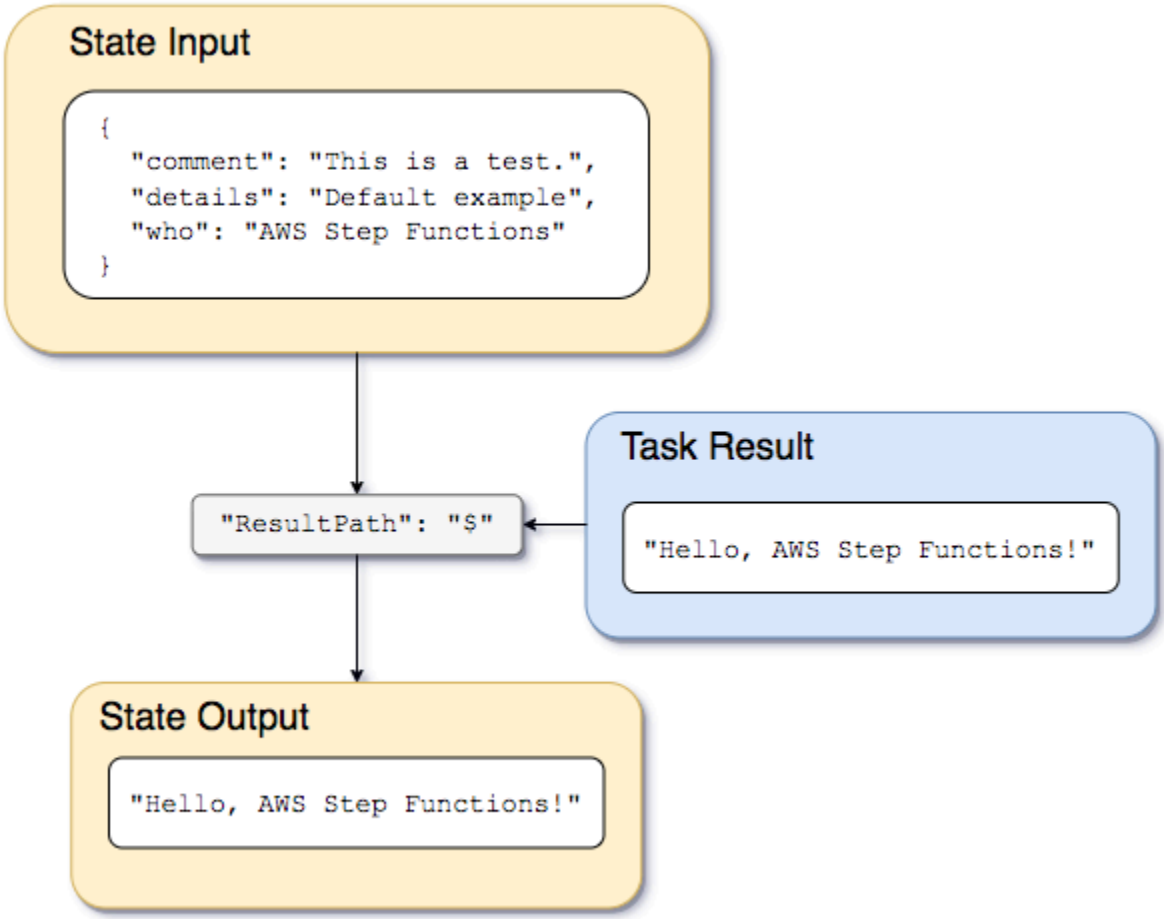
- Usa ResultPath per sostituire l'input con il risultato
- Eliminare il risultato e mantenere l'input originale
- Usa ResultPath per includere il risultato nell'input
- Usa ResultPath per aggiornare un nodo nell'input con il risultato
- ResultPath Da utilizzare per includere sia l'errore che l'input in un Catch

> **Tip**
>
> Usa il simulatore di flusso di dati nella console Step Functions per testare la sintassi del percorso JSON, comprendere meglio come i dati vengono manipolati all'interno di uno stato e vedere come i dati vengono trasferiti tra gli stati.

Usa ResultPath per sostituire l'input con il risultato

Se non specifichi `ResultPath`, il comportamento predefinito è lo stesso comportamento ottenuto specificando `"ResultPath": "$"`. Poiché questo indica allo stato di sostituire l'intero input con il risultato, l'input dello stato viene completamente sostituito con il risultato proveniente dal task.

Il diagramma seguente mostra come `ResultPath` può sostituire completamente l'input con il risultato del task.



Utilizza la macchina a stati e la funzione Lambda descritte in Creazione di una macchina a stati Step Functions che utilizza Lambda e modifica il tipo di integrazione del servizio in integrazione AWS SDK per la funzione Lambda. A tale scopo, specifica la funzione Lambda Amazon Resource Name (ARN) nel `Resource` campo dello `Task` stato, come mostrato nell'esempio seguente. L'utilizzo dell'integrazione AWS SDK garantisce che il risultato `Task` dello stato contenga solo l'output della funzione Lambda senza metadati.

```
{
   "StartAt":"CallFunction",
   "States":{
      "CallFunction": {
         "Type":"Task",
         "Resource":"arn:aws:lambda:us-east-2:123456789012:function:HelloFunction",
         "End": true
      }
   }
```

```
}
```

Quindi, trasmetti il seguente input:

```
{
  "comment": "This is a test of the input and output of a Task state.",
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

La funzione Lambda fornisce il seguente risultato.

```
"Hello, AWS Step Functions!"
```

> **Tip**
>
> È possibile visualizzare questo risultato sulla Step Functionsconsole. Per fare ciò, nella pagina Dettagli di esecuzione della console, scegli la Lambda funzione nella vista Grafico. Quindi, scegli la scheda Output nel Dettagli del passaggio riquadro per vedere questo risultato.

Se `ResultPath` non è specificato nello stato, o se `"ResultPath": "$"` è impostato, l'input dello stato viene sostituito dal risultato della funzione Lambda e l'output dello stato è il seguente.

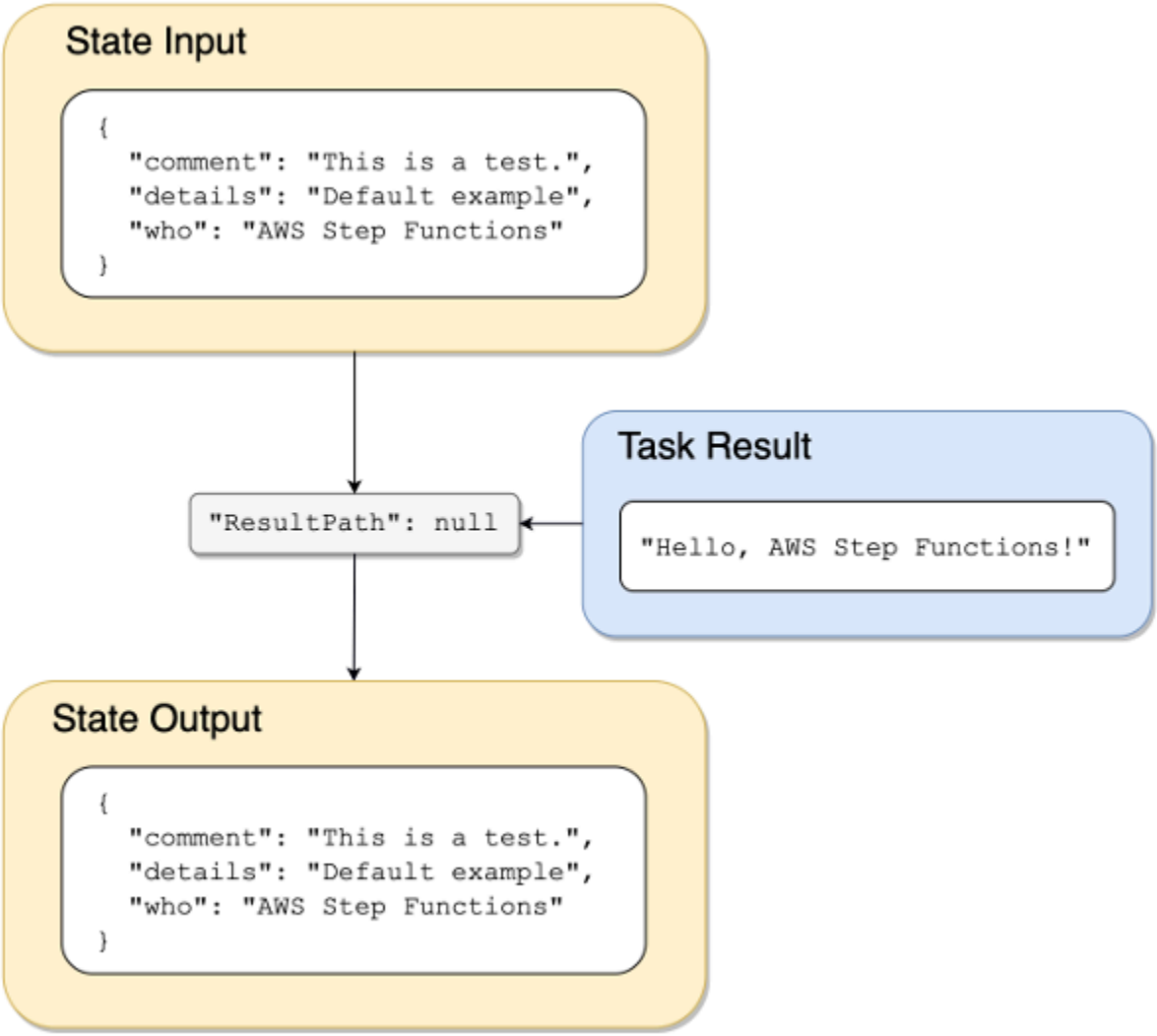
```
"Hello, AWS Step Functions!"
```

> **Note**
>
> `ResultPath` viene utilizzato per includere il contenuto del risultato con l'input prima di passarlo all'output. Se `ResultPath` non viene specificato, tuttavia, per default viene sostituito l'intero input.

Eliminare il risultato e mantenere l'input originale

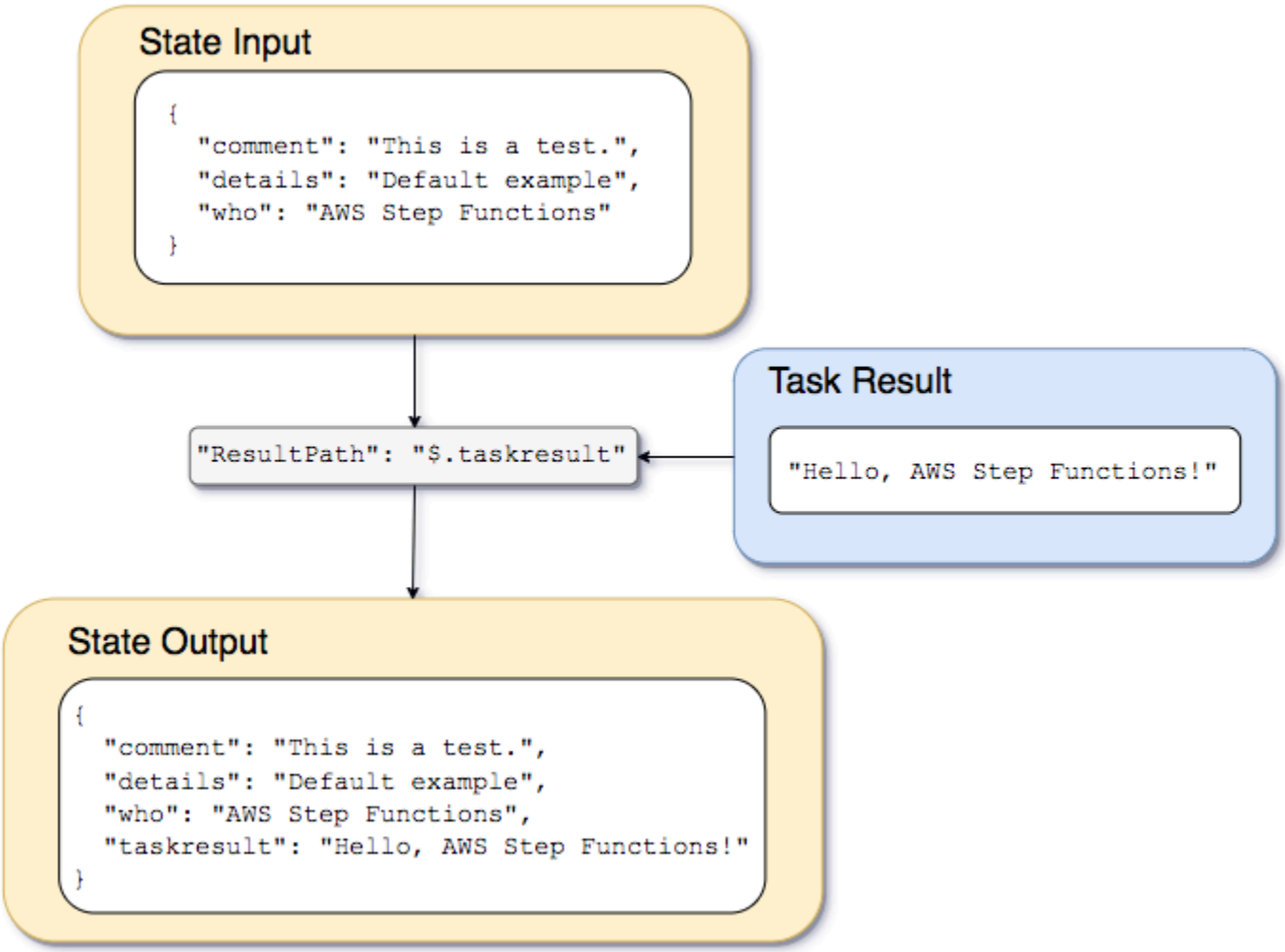
Se si imposta `ResultPath` su `null`, l'input originale verrà trasferito all'output. Utilizzando `"ResultPath": null`, il payload di input dello stato verrà copiato direttamente nell'output, indipendentemente dal risultato.

Il diagramma seguente mostra come il valore null di `ResultPath` copia l'input direttamente nell'output.



Usa ResultPath per includere il risultato nell'input

Il diagramma seguente mostra come `ResultPath` può includere il risultato con l'input.



Utilizzando la macchina a stati e la funzione Lambda descritte nel Creazione di una macchina a stati Step Functions che utilizza Lambda tutorial, potremmo passare il seguente input.

```
{
  "comment": "This is a test of the input and output of a Task state.",
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

Il risultato della funzione Lambda è il seguente.

```
"Hello, AWS Step Functions!"
```

Per preservare l'input, inserire il risultato della funzione Lambda e quindi passare il JSON combinato allo stato successivo, potremmo `ResultPath` impostare quanto segue.

```
"ResultPath": "$.taskresult"
```

Ciò include il risultato della funzione Lambda con l'input originale.

```
{
  "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!"
}
```

L'output della funzione Lambda viene aggiunto all'input originale come valore per. `taskresult` L'input, incluso il valore appena inserito, viene passato allo stato successivo.

È anche possibile inserire il risultato in un nodo figlio dell'input. Impostare `ResultPath` come segue.

```
"ResultPath": "$.strings.lambdaresult"
```

Avvia un'esecuzione utilizzando i seguenti input.

```
{
  "comment": "An input comment.",
  "strings": {
    "string1": "foo",
    "string2": "bar",
    "string3": "baz"
  },
  "who": "AWS Step Functions"
}
```

Il risultato della funzione Lambda viene inserito come figlio del `strings` nodo nell'input.

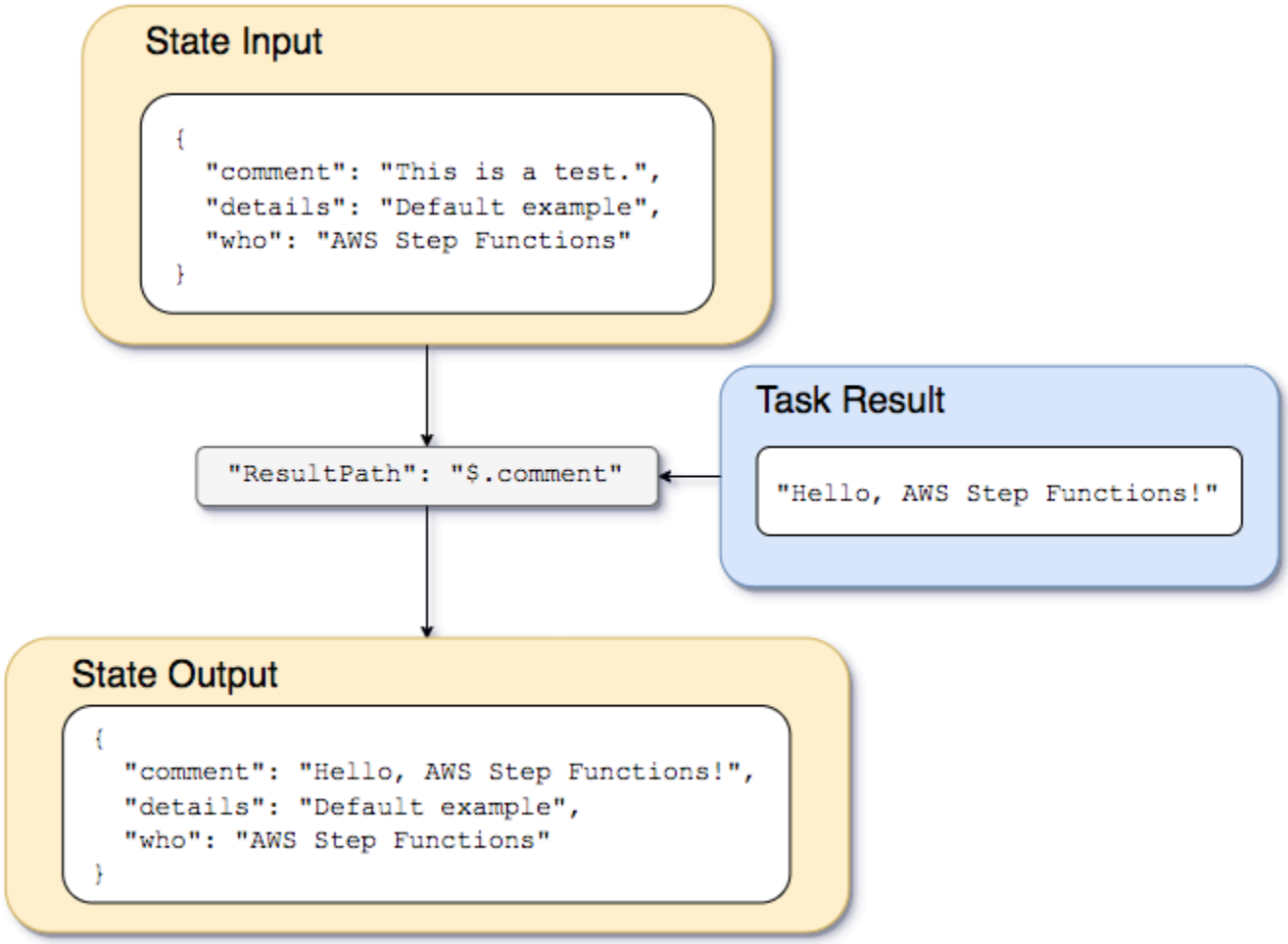
```
{
  "comment": "An input comment.",
  "strings": {
    "string1": "foo",
    "string2": "bar",
    "string3": "baz",
```

```
    "lambdaresult": "Hello, AWS Step Functions!"
  },
  "who": "AWS Step Functions"
}
```

L'output dello stato includerà ora il testo JSON di input originale con il risultato come nodo figlio.

Usa ResultPath per aggiornare un nodo nell'input con il risultato

Il diagramma seguente mostra come `ResultPath` può aggiornare il valore dei nodi JSON esistenti nell'input con i valori del risultato del task.



Utilizzando l'esempio della macchina a stati e della funzione Lambda descritta nel Creazione di una macchina a stati Step Functions che utilizza Lambda tutorial, potremmo passare il seguente input.

```
{
```

```
  "comment": "This is a test of the input and output of a Task state.",
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

Il risultato della funzione Lambda è il seguente.

```
Hello, AWS Step Functions!
```

Invece di conservare l'input e inserire il risultato come nuovo nodo nel testo JSON, possiamo sovrascrivere un nodo esistente.

Così come l'omissione o l'impostazione di `"ResultPath": "$"` determina la sovrascrittura dell'intero nodo, è possibile ad esempio specificare un singolo nodo da sovrascrivere con il risultato.

```
"ResultPath": "$.comment"
```

Poiché il `comment` nodo esiste già nell'input dello stato, l'impostazione `ResultPath` su `"$.comment"` sostituisce quel nodo nell'input con il risultato della funzione Lambda. Senza applicare ulteriori filtri tramite `OutputPath`, all'output viene passato quanto segue.

```
{
  "comment": "Hello, AWS Step Functions!",
  "details": "Default behavior example",
  "who": "AWS Step Functions",
}
```

Il valore per il `comment` nodo`"This is a test of the input and output of a Task state."`,, viene sostituito dal risultato della funzione Lambda: `"Hello, AWS Step Functions!"` nello state output.

ResultPath Da utilizzare per includere sia l'errore che l'input in un **`Catch`**

Il tutorial Gestione delle condizioni di errore utilizzando una macchina a stati Step Functions illustra come utilizzare una macchina a stati per rilevare un errore. In alcuni casi, può essere opportuno conservare l'input originale con l'errore. Per includere l'errore con l'input originale, anziché sostituirlo, utilizza `ResultPath` in un'istruzione `Catch`.

```
"Catch": [{
```

```
  "ErrorEquals": ["States.ALL"],
  "Next": "NextTask",
  "ResultPath": "$.error"
}]
```

Se rileva un errore, l'istruzione `Catch` precedente include il risultato in un nodo `error` all'interno dell'input dello stato. Ad esempio, con l'input seguente:

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

l'output dello stato in caso di rilevamento di un errore sarà il seguente.

```
{
  "foo": "bar",
  "error": {
    "Error": "Error here"
  }
}
```

Per ulteriori informazioni sulla gestione degli errori, consulta i seguenti argomenti:

- Gestione degli errori in Step Functions
- Gestione delle condizioni di errore utilizzando una macchina a stati Step Functions

OutputPath

`OutputPath` consente di selezionare una parte dell'output dello stato da passare allo stato successivo. In questo modo sarà possibile filtrare le informazioni indesiderate e trasferire solo la parte di JSON richiesta.

Se non si specifica un `OutputPath`, il valore di default è `$`. Questo trasferisce l'intero nodo JSON (stabilito dallo stato dell'input, il risultato dell'attività e `ResultPath`) allo stato successivo.

> **Tip**
>
> Usa il simulatore di flusso di dati nella console Step Functions per testare la sintassi del percorso JSON, comprendere meglio come i dati vengono manipolati all'interno di uno stato e vedere come i dati vengono trasferiti tra gli stati.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Percorsi nella lingua degli Stati dell'Amazzonia
- InputPathResultPath, ed OutputPath esempi
- Passa JSON statico come parametri
- Elaborazione di input e output in Step Functions

InputPathResultPath, ed OutputPath esempi

Qualsiasi stato diverso da uno Fail stato o da uno Succeed stato può includere i campi di elaborazione di input e output, ad esempio`InputPath`,`ResultPath`, o`OutputPath`. Inoltre, gli Choice stati Attendi e non supportano il `ResultPath` campo. Con questi campi, puoi usare JsonPatha per filtrare i dati JSON durante il flusso di lavoro.

Puoi anche utilizzare il `Parameters` campo per manipolare i dati JSON durante il flusso di lavoro. Per ulteriori informazioni sull'uso di `Parameters`, consultare InputPath, Parametri e ResultSelector.

Comincia ad esempio con la funzione e la macchina a stati AWS Lambda descritte nel tutorial Creazione di una macchina a stati Step Functions che utilizza Lambda. Modifica la macchina a stati affinché includa i seguenti `InputPath`, `ResultPath` e `OutputPath`.

```
{
  "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
    }
  }
}
```

Avvia un'esecuzione utilizzando i seguenti input.

```
{
```

```
  "comment": "An input comment.",
  "data": {
    "val1": 23,
    "val2": 17
  },
  "extra": "foo",
  "lambda": {
    "who": "AWS Step Functions"
  }
}
```

Supponiamo che i `extra` nodi `comment` and possano essere scartati, ma che vogliamo includere l'output della funzione Lambda e preservare le informazioni nel nodo. `data`

Nella macchina a stati aggiornata, lo stato `Task` viene modificato in modo da elaborare l'input al task.

```
"InputPath": "$.lambda",
```

Questa riga nella definizione della macchina a stati limita l'input del task al nodo `lambda` dall'input dello stato. La funzione Lambda riceve solo l'oggetto JSON `{"who": "AWS Step Functions"}` come input.

```
"ResultPath": "$.data.lambdaresult",
```

Questo `ResultPath` indica alla macchina a stati di inserire il risultato della funzione Lambda in un nodo denominato `lambdaresult` figlio del `data` nodo nell'input della macchina a stati originale. Poiché non stiamo eseguendo nessun'altra manipolazione sull'input originale e sul risultato utilizzando il risultato`OutputPath`, l'output dello stato ora include il risultato della funzione Lambda con l'input originale.

```
{
  "comment": "An input comment.",
  "data": {
    "val1": 23,
    "val2": 17,
    "lambdaresult": "Hello, AWS Step Functions!"
  },
  "extra": "foo",
  "lambda": {
    "who": "AWS Step Functions"
```

```
  }
}
```

Tuttavia, il nostro obiettivo era preservare solo il `data` nodo e includere il risultato della funzione Lambda. `OutputPath`filtra questo JSON combinato prima di passarlo all'output di stato.

```
"OutputPath": "$.data",
```

Questo seleziona solo il nodo `data` dall'input originale (tra cui il figlio `lambdaresult` inserito da `ResultPath`) da trasmettere all'output. L'output dello stato viene filtrato come segue.

```
{
  "val1": 23,
  "val2": 17,
  "lambdaresult": "Hello, AWS Step Functions!"
}
```

In questo stato `Task`:

1. `InputPath`invia solo il `lambda` nodo dall'input alla funzione Lambda.
2. `ResultPath` inserisce il risultato come figlio del nodo `data` nell'input originale.
3. `OutputPath`filtra l'input di stato (che ora include il risultato della funzione Lambda) in modo che passi solo il `data` nodo all'output di stato.

Example per manipolare l'input, il risultato e l'output finale della macchina a stato originale usando JsonPath

Considera la seguente macchina statale che verifica l'identità e l'indirizzo di un richiedente l'assicurazione.

> **Note**
>
> Per visualizzare l'esempio completo, vedere Come utilizzare 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
            }
          }
        }
      ]
    }
  }
}
```

Se esegui questa macchina a stati utilizzando il seguente input, l'esecuzione fallisce perché le funzioni Lambda che eseguono la verifica prevedono solo i dati che devono essere verificati come input. Pertanto, è necessario specificare i nodi che contengono le informazioni da verificare utilizzando un appropriatoJsonPath.

```
{
  "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
      },
    ]
  }
}
```

Per specificare il nodo che la funzione *`check-identity`* Lambda deve utilizzare, usa il `InputPath` campo come segue:

```
"InputPath": "$.data.identity"
```

E per specificare il nodo che la funzione *`check-address`* Lambda deve usare, usa il `InputPath` campo come segue:

```
"InputPath": "$.data.address"
```

Ora, se desideri memorizzare il risultato della verifica all'interno dell'input della macchina a stato originale, usa il `ResultPath` campo come segue:

```
"ResultPath": "$.results"
```

Tuttavia, se hai bisogno solo dei risultati dell'identità e della verifica ed elimini l'input originale, utilizza il `OutputPath` campo come segue:

```
"OutputPath": "$.results"
```

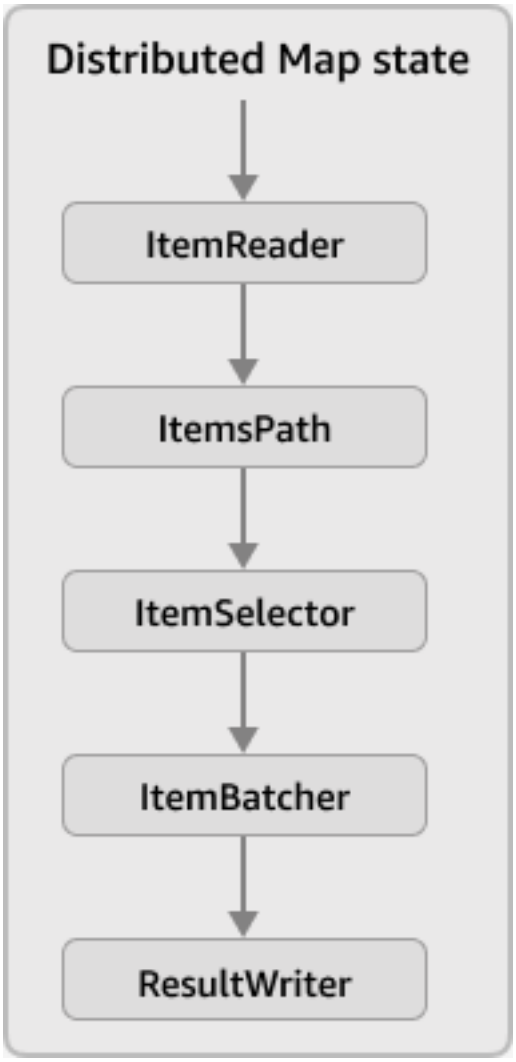
Per ulteriori informazioni, consulta Elaborazione di input e output in Step Functions.

Campi di input e output dello stato della mappa

`Map`gli stati eseguono contemporaneamente iterazioni su una raccolta di elementi in un set di dati, come un array JSON, un elenco di oggetti Amazon S3 o le righe di un file CSV in un bucket Amazon S3. Ripete una serie di passaggi per ogni elemento della collezione. È possibile configurare l'input che lo `Map` stato riceve e l'output che genera utilizzando questi campi. Step Functions applica ogni campo nello stato della mappa distribuita nell'ordine mostrato nell'elenco e nell'illustrazione seguenti:

> **Note**
>
> In base al tuo caso d'uso, potresti non aver bisogno di applicare tutti questi campi.

1. ItemReader
2. ItemsPath
3. ItemSelector
4. ItemBatcher
5. ResultWriter



> **Note**
>
> Questi campi di input e output di `Map` stato non sono attualmente disponibili nel simulatore di flusso di dati nella console Step Functions.

ItemReader

Il`ItemReader`field è un oggetto JSON, che specifica un set di dati e la sua posizione. UNStato della mappa distribuitautilizza questo set di dati come input. L'esempio seguente mostra la sintassi della`ItemReader`campo se il tuo set di dati è un file CSV archiviato in un bucket Amazon S3.

```
"ItemReader": {
    "ReaderConfig": {
```

```
        "InputType": "CSV",
        "CSVHeaderLocation": "FIRST_ROW"
    },
    "Resource": "arn:aws:states:::s3:getObject",
    "Parameters": {
        "Bucket": "myBucket",
        "Key": "csvDataset/ratings.csv"
    }
}
```

> **Tip**
>
> In Workflow Studio, specifichi il set di dati e la sua posizione nelFonte dell'articolocampo.

Indice

- Contenuto del ItemReader campo
- Esempi di set di dati
- Politiche IAM per set di dati

Contenuto del ItemReader campo

A seconda del set di dati, il contenuto del`ItemReader`il campo varia. Ad esempio, se il set di dati è un array JSON passato da una fase precedente del flusso di lavoro,`ItemReader`il campo è omesso. Se il set di dati è un'origine dati Amazon S3, questo campo contiene i seguenti sottocampi.

`ReaderConfig`

Un oggetto JSON che specifica i seguenti dettagli:

- `InputType`

  Speciifica il tipo di origine dati Amazon S3, ad esempio file CSV, oggetto, file JSON o un elenco di inventario Amazon S3. In Workflow Studio, puoi selezionare un tipo di input dalFonte dell'articolo Amazon S3elenco a discesa sotto ilFonte dell'articolocampo.

- `CSVHeaderLocation`

> **Note**
>
> È necessario specificare questo campo solo se si utilizza un file CSV come set di dati.

Accetta uno dei seguenti valori per specificare la posizione dell'intestazione della colonna:

> **Important**
>
> Attualmente, Step Functions supporta intestazioni CSV fino a 10 KB.

- `FIRST_ROW`— Usa questa opzione se la prima riga del file è l'intestazione.
- `GIVEN`— Utilizzate questa opzione per specificare l'intestazione all'interno della definizione della macchina a stati. Ad esempio, se il file CSV contiene i seguenti dati.

```
1,307,3.5,1256677221
1,481,3.5,1256677456
1,1091,1.5,1256677471
...
```

Fornisci il seguente array JSON come intestazione CSV.

```
"ItemReader": {
    "ReaderConfig": {
        "InputType": "CSV",
        "CSVHeaderLocation": "GIVEN",
        "CSVHeaders": [
            "userId",
            "movieId",
            "rating",
            "timestamp"
        ]
    }
}
```

> **Tip**
>
> In Workflow Studio, puoi trovare questa opzione inConfigurazione aggiuntivanelFonte dell'articolocampo.

- `MaxItems`

  Limita il numero di elementi di dati trasmessi a`Map`stato. Ad esempio, supponiamo di fornire un file CSV contenente 1000 righe e di specificare un limite di 100. Quindi, l'interprete passasolomancano 100 righe al`Map`stato. La`Map`state elabora gli elementi in ordine sequenziale, a partire dalla riga di intestazione.

  Per impostazione predefinita,`Map`lo stato itera su tutti gli elementi del set di dati specificato.

  > **Note**
  >
  > Attualmente, è possibile specificare un limite fino a 100.000.000. IlStato della mappa distribuitainterrompe la lettura degli elementi che superano questo limite.

  > **Tip**
  >
  > In Workflow Studio, puoi trovare questa opzione inConfigurazione aggiuntivanelFonte dell'articolocampo.

  In alternativa, è possibile specificare unpercorso di riferimentoa una coppia chiave-valore esistente nelstato della mappa distribuitainput. Questo percorso deve risolversi in un numero intero positivo. Specificate il percorso di riferimento nel`MaxItemsPath`sottocampo.

  > **Important**
  >
  > È possibile specificare una delle`MaxItems`o`MaxItemsPath`sottocampo, ma non entrambi.

`Resource`

L'azione dell'API Amazon S3 che Step Functions deve richiamare a seconda del set di dati specificato.

`Parameters`

Un oggetto JSON che specifica il nome del bucket Amazon S3 e la chiave dell'oggetto in cui è archiviato il set di dati.

> **Important**
>
> Assicurati che i tuoi bucket Amazon S3 siano sotto gli stessiAccount AWSeRegione AWScome la tua macchina di stato.

Esempi di set di dati

È possibile specificare una delle seguenti opzioni come set di dati:

- array JSON di un passaggio precedente
- Un elenco di oggetti Amazon S3
- File JSON in un bucket Amazon S3
- File CSV in un bucket Amazon S3
- Elenco inventario Amazon S3

> **Important**
>
> Step Functions necessita di autorizzazioni appropriate per accedere ai set di dati Amazon S3 che utilizzi. Per informazioni sulle politiche IAM per i set di dati, consultaPolitiche IAM per set di dati.

Array JSON di un passaggio precedente

UNStato della mappa distribuitapuò accettare un input JSON della fase precedente nel flusso di lavoro. Questo input deve essere un array o deve contenere un array all'interno di un nodo specifico. Per selezionare un nodo che contiene l'array, è possibile utilizzare`ItemsPath`campo.

Per elaborare i singoli elementi dell'array,Stato della mappa distribuitaavvia l'esecuzione di un workflow secondario per ogni elemento dell'array. Le schede seguenti mostrano esempi dell'input passato a`Map`stato e l'input corrispondente all'esecuzione di un workflow secondario.

> **Note**
>
> Step Functions omette`ItemReader`campo quando il set di dati è un array JSON di un passaggio precedente.

Input passed to the Map state

Considera il seguente array JSON di tre elementi.

```
"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

LaStato della mappa distribuitaavvia tre esecuzioni di workflow secondarie. Ogni esecuzione riceve un elemento dell'array come input. L'esempio seguente mostra l'input ricevuto dall'esecuzione di un workflow secondario.

```
{
  "verdict": "true",
  "statement_date": "6/11/2008",
```

```
  "statement_source": "speech"
}
```

Esempio di oggetti Amazon S3

UNStato della mappa distribuitapuò iterare sugli oggetti archiviati in un bucket Amazon S3. Quando l'esecuzione del flusso di lavoro raggiunge il`Map`state, Step Functions invoca il[ListObjectsV2](#)Azione API, che restituisce un array di metadati degli oggetti Amazon S3. In questo array, ogni elemento contiene dati, comeETageChiave, per i dati memorizzati nel bucket.

Per elaborare singoli elementi dell'array,Stato della mappa distribuitaavvia l'esecuzione di un workflow secondario. Ad esempio, supponiamo che il bucket Amazon S3 contenga 100 immagini. Quindi, l'array è tornato dopo aver richiamato il`ListObjectsV2`L'azione API contiene 100 elementi. IlStato della mappa distribuitaavvia quindi 100 esecuzioni di workflow secondarie per elaborare ogni elemento dell'array.

> **Note**
>
> - Attualmente, Step Functions include anche un elemento per ogni cartella creata in un bucket Amazon S3 utilizzando la console Amazon S3. Ciò si traduce in un'esecuzione aggiuntiva del flusso di lavoro secondario avviata daStato della mappa distribuita. Per evitare di creare un'esecuzione aggiuntiva del flusso di lavoro secondario per la cartella, si consiglia di utilizzare ilAWS CLIper creare cartelle. Per ulteriori informazioni, consulta la pagina[Comandi Amazon S3 di alto livello](#)nelAWS Command Line InterfaceGuida per l'utente.
> - Step Functions necessita di autorizzazioni appropriate per accedere ai set di dati Amazon S3 che utilizzi. Per informazioni sulle politiche IAM per i set di dati, consulta[Politiche IAM per set di dati](#).

Le schede seguenti mostrano esempi di`ItemReader`la sintassi del campo e l'input passato all'esecuzione di un workflow secondario per questo set di dati.

ItemReader syntax

In questo esempio, hai organizzato i dati, che includono immagini, file JSON e oggetti, all'interno di un prefisso denominato`processData`in un bucket Amazon S3 denominato`myBucket`.

```
"ItemReader": {
    "Resource": "arn:aws:states:::s3:listObjectsV2",
    "Parameters": {
        "Bucket": "myBucket",
        "Prefix": "processData"
    }
}
```

Input passed to a child workflow execution

LaStato della mappa distribuitaavvia tante esecuzioni di workflow secondarie quanti sono gli elementi presenti nel bucket Amazon S3. L'esempio seguente mostra l'input ricevuto dall'esecuzione di un flusso di lavoro secondario.

```
{
  "Etag": "\"05704fbdccb224cb01c59005bebbad28\"",
  "Key": "processData/images/n02085620_1073.jpg",
  "LastModified": 1668699881,
  "Size": 34910,
  "StorageClass": "STANDARD"
}
```

File JSON in un bucket Amazon S3

UNStato della mappa distribuitaè in grado di accettare un file JSON archiviato in un bucket Amazon S3 come set di dati. Il file JSON deve contenere un array.

Quando l'esecuzione del flusso di lavoro raggiunge il`Map`state, Step Functions invoca il[GetObject](#)Azione API per recuperare il file JSON specificato. Il`Map`state esegue quindi un'iterazione su ogni elemento dell'array e avvia un'esecuzione del flusso di lavoro secondario per ogni elemento. Ad esempio, se il file JSON contiene 1000 elementi dell'array,`Map`state avvia 1000 esecuzioni di workflow secondarie.

> **Note**
>
> - L'input di esecuzione utilizzato per avviare l'esecuzione di un workflow secondario non può superare i 256 KB. Tuttavia, Step Functions supporta la lettura di un elemento fino a 8 MB da un file CSV o JSON se poi si applica l'opzione`ItemSelector`campo per ridurre le dimensioni dell'articolo.

- Attualmente, Step Functions supporta 10 GB come dimensione massima di un singolo file in un report di inventario Amazon S3. Tuttavia, Step Functions può elaborare più di 10 GB se ogni singolo file è inferiore a 10 GB.
- Step Functions necessita di autorizzazioni appropriate per accedere ai set di dati Amazon S3 che utilizzi. Per informazioni sulle politiche IAM per i set di dati, consultaPolitiche IAM per set di dati.

Le schede seguenti mostrano esempi di`ItemReader`la sintassi del campo e l'input passato all'esecuzione di un workflow secondario per questo set di dati.

In questo esempio, immagina di avere un file JSON di nome*`factcheck.json`*. Hai memorizzato questo file all'interno di un prefisso denominato*`jsonDataset`*in un bucket Amazon S3. Di seguito è riportato un esempio del set di dati 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

LaStato della mappa distribuitaavvia tante esecuzioni di workflow secondarie quanti sono gli elementi dell'array presenti nel file JSON. L'esempio seguente mostra l'input ricevuto dall'esecuzione di un workflow secondario.

```
{
  "verdict": "true",
  "statement_date": "6/11/2008",
  "statement_source": "speech"
}
```

File CSV in un bucket Amazon S3

UNStato della mappa distribuitaè in grado di accettare un file CSV archiviato in un bucket Amazon S3 come set di dati. Se utilizzi un file CSV come set di dati, devi specificare un'intestazione di colonna CSV. Per informazioni su come specificare un'intestazione CSV, consultaContenuto del ItemReader campo.

Poiché non esiste un formato standardizzato per creare e gestire i dati nei file CSV, Step Functions analizza i file CSV in base alle seguenti regole:

- Le virgole (,) sono un delimitatore che separa i singoli campi.
- Le newline sono un delimitatore che separa i singoli record.
- I campi vengono trattati come stringhe. Per le conversioni dei tipi di dati, usa`States.StringToJson`funzione intrinseca in`ItemSelector`.
- Le virgolette doppie (» «) non sono necessarie per racchiudere le stringhe. Tuttavia, le stringhe racchiuse tra virgolette doppie possono contenere virgole e nuove righe senza che funzionino come delimitatori.
- Evita le virgolette doppie ripetendole.

- Se il numero di campi in una riga è inferiore al numero di campi nell'intestazione, Step Functions fornisce stringhe vuote per i valori mancanti.
- Se il numero di campi in una riga è superiore al numero di campi nell'intestazione, Step Functions salta i campi aggiuntivi.

Per ulteriori informazioni su come Step Functions analizza un file CSV, vedere la.Example of parsing an input CSV file.

Quando l'esecuzione del flusso di lavoro raggiunge il`Map`state, Step Functions invoca il`GetObject`Azione API per recuperare il file CSV specificato. Il`Map`state esegue quindi un'iterazione su ogni riga del file CSV e avvia un'esecuzione del flusso di lavoro secondario per elaborare gli elementi di ogni riga. Ad esempio, supponiamo di fornire un file CSV che contiene 100 righe come input. Quindi, l'interprete passa ogni riga al`Map`stato. La`Map`state elabora gli elementi in ordine seriale, a partire dalla riga di intestazione.

> **Note**
>
> - L'input di esecuzione utilizzato per avviare l'esecuzione di un workflow secondario non può superare i 256 KB. Tuttavia, Step Functions supporta la lettura di un elemento fino a 8 MB da un file CSV o JSON se poi si applica l'opzione`ItemSelector`campo per ridurre le dimensioni dell'articolo.
> - Attualmente, Step Functions supporta 10 GB come dimensione massima di un singolo file in un report di inventario Amazon S3. Tuttavia, Step Functions può elaborare più di 10 GB se ogni singolo file è inferiore a 10 GB.
> - Step Functions necessita di autorizzazioni appropriate per accedere ai set di dati Amazon S3 che utilizzi. Per informazioni sulle politiche IAM per i set di dati, consultaPolitiche IAM per set di dati.

Le schede seguenti mostrano esempi di`ItemReader`la sintassi del campo e l'input passato all'esecuzione di un workflow secondario per questo set di dati.

ItemReader syntax

Ad esempio, supponiamo di avere un file CSV di nome*`ratings.csv`*. Quindi, hai memorizzato questo file all'interno di un prefisso denominato*`csvDataset`*in un bucket 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

IlStato della mappa distribuitaavvia tante esecuzioni di workflow secondarie quante sono le righe presenti nel file CSV, esclusa la riga di intestazione, se presente nel file. L'esempio seguente mostra l'input ricevuto dall'esecuzione di un workflow secondario.

```
{
  "rating": "3.5",
  "movieId": "307",
  "userId": "1",
  "timestamp": "1256677221"
}
```

Esempio di inventario S3

UNStato della mappa distribuitaè in grado di accettare un file di inventario Amazon S3 che è archiviato in un bucket Amazon S3 come set di dati.

Quando l'esecuzione del flusso di lavoro raggiunge il`Map`state, Step Functions invoca ilGetObjectAzione API per recuperare il file manifesto dell'inventario Amazon S3 specificato. Il`Map`state esegue quindi un'iterazione sugli oggetti nell'inventario per restituire una serie di metadati degli oggetti di inventario Amazon S3.

> **Note**
>
> - Attualmente, Step Functions supporta 10 GB come dimensione massima di un singolo file in un report di inventario Amazon S3. Tuttavia, Step Functions può elaborare più di 10 GB se ogni singolo file è inferiore a 10 GB.
> - Step Functions necessita di autorizzazioni appropriate per accedere ai set di dati Amazon S3 che utilizzi. Per informazioni sulle politiche IAM per i set di dati, consultaPolitiche IAM per set di dati.

Di seguito è riportato un esempio di modello di inventario in formato CSV. Questo file include gli oggetti denominaticsvDataseteimageDataset, che sono archiviati in un bucket Amazon S3 denominatosourceBucket.

```
"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**
>
> Attualmente, Step Functions non supporta i report di inventario Amazon S3 definiti dall'utente come set di dati. È inoltre necessario assicurarsi che il formato di output del report di inventario di Amazon S3 sia CSV. Per ulteriori informazioni sugli inventari Amazon S3 e su come configurarli, consulta la.Inventario Amazon S3nelGuida per l'utente di Amazon S3.

Il seguente esempio di file manifesto dell'inventario mostra le intestazioni CSV per i metadati degli oggetti di inventario.

```
{
  "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"
  } ]
}
```

Le schede seguenti mostrano esempi di`ItemReader`la sintassi del campo e l'input passato all'esecuzione di un workflow secondario per questo set di dati.

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"
}
```

A seconda dei campi selezionati durante la configurazione del report sull'inventario di Amazon S3, il contenuto del`manifest.json`il file può variare rispetto all'esempio mostrato.

Politiche IAM per set di dati

Quando crei flussi di lavoro con la console Step Functions, Step Functions può generare automaticamente policy IAM basate sulle risorse nella definizione del flusso di lavoro. Queste politiche includono i privilegi minimi necessari per consentire al ruolo della macchina a stati di richiamare il`StartExecution`Azione API perStato della mappa distribuita. Queste politiche includono anche i privilegi minimi necessari per accedere a Step Functions.AWSrisorse, come bucket e oggetti Amazon S3 e funzioni Lambda. Ti consigliamo vivamente di includere solo le autorizzazioni necessarie nelle tue policy IAM. Ad esempio, se il tuo flusso di lavoro include un`Map`imposta in modalità distribuita, limita le tue policy allo specifico bucket e alla cartella Amazon S3 che contiene il set di dati.

> **Important**
>
> Se specifichi un bucket Amazon S3 e un oggetto, o prefisso, con unpercorso di riferimentoa una coppia chiave-valore esistente nelstato della mappa distribuitainput, assicurati di aggiornare le policy IAM per il tuo flusso di lavoro. Ambita le policy fino al bucket e ai nomi degli oggetti in cui il percorso si risolve in fase di esecuzione.

I seguenti esempi di policy IAM concedono i privilegi minimi necessari per accedere ai set di dati Amazon S3 utilizzandoListObjectsV2eGetObjectAzioni API.

Example Policy IAM per oggetti Amazon S3 come set di dati

L'esempio seguente mostra una policy IAM che concede i privilegi minimi per accedere agli oggetti organizzati all'interno*`processImages`*in un bucket Amazon S3 denominato*`myBucket`*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket"
            ],
            "Condition": {
```

```
                "StringLike": {
                    "s3:prefix": [
                        "processImages"
                    ]
                }
            }
        }
    ]
}
```

Example Politica IAM per un file CSV come set di dati

L'esempio seguente mostra una policy IAM che concede i privilegi minimi per accedere a un file CSV denominato*ratings.csv*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket/csvDataset/ratings.csv"
            ]
        }
    ]
}
```

Example Policy IAM per un inventario di Amazon S3 come set di dati

L'esempio seguente mostra una policy IAM che concede i privilegi minimi per accedere a un report di inventario di 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

Usa il `ItemsPath` campo per selezionare un array all'interno di un input JSON fornito a uno `Map` stato. Lo `Map` stato ripete una serie di passaggi per ogni elemento dell'array. Per impostazione predefinita, lo `Map` stato è `ItemsPath` impostato su$, che seleziona l'intero input. Se l'input allo `Map` stato è un array JSON, esegue un'iterazione per ogni elemento dell'array, passando quell'elemento all'iterazione come input.

> **Note**
>
> È possibile `ItemsPath` utilizzarlo nello stato Mappa distribuita solo se si utilizza un input JSON passato da uno stato precedente nel flusso di lavoro.

È possibile utilizzare il `ItemsPath` campo per specificare una posizione nell'input che punti all'array JSON utilizzato per le iterazioni. Il valore di `ItemsPath` deve essere un percorso di riferimento e quel percorso deve puntare all'array JSON. Ad esempio, considera l'input a uno stato `Map` che include due matrici, come nell'esempio seguente.

```
{
  "ThingsPiratesSay": [
    {
      "say": "Avast!"
    },
    {
      "say": "Yar!"
    },
    {
      "say": "Walk the Plank!"
    }
  ],
  "ThingsGiantsSay": [
```

```
    {
      "say": "Fee!"
    },
    {
      "say": "Fi!"
    },
    {
      "say": "Fo!"
    },
    {
      "say": "Fum!"
    }
  ]
}
```

In questo caso, puoi specificare quale array utilizzare per le iterazioni di `Map` stato selezionandolo con`ItemsPath`. La seguente definizione della macchina a stati specifica l'`ThingsPiratesSay`array nell'input `ItemsPath` utilizzando.Quindi esegue un'iterazione dello stato di `SayWord` passaggio per ogni elemento dell'array. `ThingsPiratesSay`

```
{
  "StartAt": "PiratesSay",
  "States": {
    "PiratesSay": {
      "Type": "Map",
      "ItemsPath": "$.ThingsPiratesSay",
      "ItemProcessor": {
         "StartAt": "SayWord",
         "States": {
           "SayWord": {
             "Type": "Pass",
             "End": true
           }
         }
      },
      "End": true
    }
  }
}
```

Durante l'elaborazione dell'input, lo `Map` stato viene applicato `ItemsPath` dopo `InputPath`. Funziona sull'ingresso effettivo allo stato dopo aver `InputPath` filtrato l'ingresso.

Per ulteriori informazioni sugli stati Map, consulta quanto segue.

- Stato della mappa
- Modalità di elaborazione dello stato della mappa
- Ripeti un'azione utilizzando lo stato della mappa in linea
- Elaborazione di input e Map output in stato di linea

ItemSelector

Per impostazione predefinita, l'input effettivo per lo Map stato è l'insieme di singoli elementi di dati presenti nell'input in stato grezzo. Il ItemSelector campo consente di sovrascrivere i valori degli elementi di dati prima che vengano trasmessi allo stato. Map Per sovrascrivere i valori, specifica un input JSON valido che contenga una raccolta di coppie chiave-valore. Queste coppie possono essere valori statici forniti nella definizione della macchina a stati, valori selezionati dall'input di stato utilizzando un percorso o valori a cui si accede dall'oggetto di contesto.

Se specificate coppie chiave-valore utilizzando un percorso o un oggetto di contesto, il nome della chiave deve terminare con. .$

> **Note**
>
> Il ItemSelector campo sostituisce il Parameters campo all'interno dello Map stato. Se utilizzi il Parameters campo nelle definizioni dei tuoi Map stati per creare input personalizzati, ti consigliamo vivamente di sostituirli conItemSelector.

È possibile specificare il ItemSelector campo sia nello stato della mappa in linea che in quello della mappa distribuita.

Ad esempio, considera il seguente input JSON che contiene una matrice di tre elementi all'interno del imageData nodo. Per ogni iterazione di *Map* stato, un elemento dell'array viene passato all'iterazione come input.

```
[
  {
    "resize": "true",
    "format": "jpg"
```

```
  },
  {
    "resize": "false",
    "format": "png"
  },
  {
    "resize": "true",
    "format": "jpg"
  }
]
```

Utilizzando il `ItemSelector` campo, è possibile definire un input JSON personalizzato per sovrascrivere l'input originale, come mostrato nell'esempio seguente. Step Functions passa quindi questo input personalizzato a ciascuna iterazione di *`Map`* stato. L'input personalizzato contiene un valore statico per `size` i dati di un oggetto di contesto per `Map` lo stato. L'oggetto di `$$.Map.Item.Value` contesto contiene il valore di ogni singolo elemento di dati.

```
{
  "ItemSelector": {
    "size": 10,
    "value.$": "$$.Map.Item.Value"
  }
}
```

L'esempio seguente mostra l'input ricevuto da un'iterazione dello stato della mappa in linea:

```
{
  "size": 10,
  "value": {
    "resize": "true",
    "format": "jpg"
  }
}
```

> **Tip**
>
> Per un esempio completo di uno stato di una mappa distribuita che utilizza il `ItemSelector` campo, vedereGuida introduttiva all'utilizzo dello stato di Distributed Map.

ItemBatcher

Il `ItemBatcher` campo è un oggetto JSON, che specifica di elaborare un gruppo di elementi nell'esecuzione di un singolo flusso di lavoro secondario. Usa il batch per elaborare file CSV o array JSON di grandi dimensioni o set di oggetti Amazon S3 di grandi dimensioni.

L'esempio seguente mostra la sintassi del `ItemBatcher` campo. Nella sintassi seguente, il numero massimo di elementi che ogni esecuzione del flusso di lavoro secondario deve elaborare è impostato su 100.

```
{
  "ItemBatcher": {
    "MaxItemsPerBatch": 100
  }
}
```

Per impostazione predefinita, ogni elemento di un set di dati viene passato come input alle singole esecuzioni del flusso di lavoro secondario. Ad esempio, supponiamo di specificare un file JSON come input che contiene il seguente array:

```
[
  {
    "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"
  },
  ...
]
```

Per l'input specificato, ogni esecuzione del flusso di lavoro secondario riceve un elemento dell'array come input. L'esempio seguente mostra l'input dell'esecuzione di un workflow secondario:

```
{
  "verdict": "true",
  "statement_date": "6/11/2008",
  "statement_source": "speech"
}
```

Per ottimizzare le prestazioni e i costi del processo di elaborazione, selezionate una dimensione del batch che bilanci il numero di articoli con il tempo di elaborazione degli articoli. Se si utilizza il batch, Step Functions aggiunge gli elementi a un array Items. Quindi passa l'array come input a ogni esecuzione del flusso di lavoro secondario. L'esempio seguente mostra un batch di due elementi passati come input all'esecuzione di un flusso di lavoro secondario:

```
{
  "Items": [
    {
      "verdict": "true",
      "statement_date": "6/11/2008",
      "statement_source": "speech"
    },
    {
      "verdict": "false",
      "statement_date": "6/7/2022",
      "statement_source": "television"
    }
  ]
}
```

Tip

Per saperne di più sull'uso del `ItemBatcher` campo nei tuoi flussi di lavoro, prova i seguenti tutorial e workshop:

- Elabora un intero batch di dati all'interno di una funzione Lambda
- Esegui iterazioni sugli elementi di un batch all'interno delle esecuzioni del flusso di lavoro secondario
- Parallelizzazione su larga scala con mappa distribuita nel Modulo 14 - Elaborazione dei dati del workshop AWS Step Functions

Indice

- Campi per specificare il batch degli articoli

Campi per specificare il batch degli articoli

Per raggruppare gli articoli, specifica il numero massimo di articoli da raggruppare, la dimensione massima del batch o entrambi. È necessario specificare uno di questi valori per raggruppare gli articoli.

Numero massimo di articoli per lotto

Specifica il numero massimo di elementi che ogni workflow secondario elabora. L'interprete limita il numero di elementi raggruppati nell'`Items`array a questo valore. Se si specificano sia il numero che la dimensione del batch, l'interprete riduce il numero di articoli in un batch per evitare di superare il limite di dimensione del batch specificato.

Se non specifichi questo valore ma fornisci un valore per la dimensione massima del batch, Step Functions elabora il maggior numero possibile di elementi in ogni esecuzione del flusso di lavoro secondario senza superare la dimensione massima del batch in byte.

Ad esempio, immagina di eseguire un'esecuzione con un file JSON di input che contiene 1130 nodi. Se si specifica un valore massimo di articoli per ogni batch di 100, Step Functions crea 12 batch. Di questi, 11 lotti contengono 100 articoli ciascuno, mentre il dodicesimo batch contiene i restanti 30 articoli.

In alternativa, è possibile specificare il numero massimo di elementi per ogni batch come percorso di riferimento a una coppia chiave-valore esistente nell'input dello stato della mappa distribuita. Questo percorso deve essere risolto in un numero intero positivo.

Ad esempio, con il seguente input:

```
{
  "maxBatchItems": 500
}
```

È possibile specificare il numero massimo di articoli da raggruppare utilizzando un percorso di riferimento come segue:

```
{
```

```
  ...
  "Map": {
    "Type": "Map",
    "MaxConcurrency": 2000,
    "ItemBatcher": {
      "MaxItemsPerBatchPath": "$.maxBatchItems"
    }
    ...
    ...
  }
}
```

> **Important**
>
> È possibile specificare il sottocampo `MaxItemsPerBatch` o il `MaxItemsPerBatchPath` sottocampo, ma non entrambi.

Numero massimo di KB per batch

Specifica la dimensione massima di un batch in byte, fino a 256 KB. Se si specificano sia un numero massimo di batch che una dimensione, Step Functions riduce il numero di articoli in un batch per evitare di superare il limite di dimensione del batch specificato.

In alternativa, è possibile specificare la dimensione massima del batch come percorso di riferimento a una coppia chiave-valore esistente nell'input dello stato della mappa distribuita. Questo percorso deve essere risolto in un numero intero positivo.

> **Note**
>
> Se si utilizza il batch e non si specifica una dimensione massima del batch, l'interprete elabora tutti gli elementi che può elaborare fino a 256 KB in ogni esecuzione del flusso di lavoro secondario.

Ad esempio, con il seguente input:

```
{
  "batchSize": 131072
}
```

È possibile specificare la dimensione massima del batch utilizzando un percorso di riferimento come segue:

```
{
  ...
  "Map": {
    "Type": "Map",
    "MaxConcurrency": 2000,
    "ItemBatcher": {
      "MaxInputBytesPerBatchPath": "$.batchSize"
    }
    ...
    ...
  }
}
```

> **Important**
>
> È possibile specificare il sottocampo `MaxInputBytesPerBatch` o il `MaxInputBytesPerBatchPath` sottocampo, ma non entrambi.

Ingresso in batch

Facoltativamente, puoi anche specificare un input JSON fisso da includere in ogni batch passato all'esecuzione di ogni flusso di lavoro secondario. Step Functions unisce questo input con l'input per ogni singola esecuzione del flusso di lavoro secondario. Ad esempio, dato il seguente inserimento fisso di una data di verifica dei fatti su una serie di elementi:

```
"ItemBatcher": {
    "BatchInput": {
        "factCheck": "December 2022"
    }
}
```

Ogni esecuzione del flusso di lavoro secondario riceve quanto segue come input:

```
{
  "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

Il *`ResultWriter`* campo è un oggetto JSON che specifica la posizione Amazon S3 in cui Step Functions scrive i risultati delle esecuzioni dei flussi di lavoro secondari avviate da uno stato di Distributed Map. Per impostazione predefinita, Step Functions non esporta questi risultati.

> **Important**
>
> Assicurati che il bucket Amazon S3 che usi per esportare i risultati di un Map Run si trovi nella stessa macchina a stati della tua Account AWS macchina a Regione AWS stati. In caso contrario, l'esecuzione della macchina a stati fallirà con l'`States.ResultWriterFailed`errore.

L'esportazione dei risultati in un bucket Amazon S3 è utile se la dimensione del payload di output supera i 256 KB. Step Functions consolida tutti i dati di esecuzione del workflow secondario, come input e output dell'esecuzione, ARN e stato di esecuzione. Quindi esporta le esecuzioni con lo stesso stato nei rispettivi file nella posizione Amazon S3 specificata. L'esempio seguente mostra la sintassi del `ResultWriter` campo se si esportano i risultati dell'esecuzione del flusso di lavoro secondario. In questo esempio, si memorizzano i risultati in un bucket denominato `myOutputBucket` all'interno di un prefisso chiamato. `csvProcessJobs`

```
{
  "ResultWriter": {
```

```
    "Resource": "arn:aws:states:::s3:putObject",
    "Parameters": {
      "Bucket": "myOutputBucket",
      "Prefix": "csvProcessJobs"
    }
  }
}
```

> **Tip**
>
> In Workflow Studio, puoi esportare i risultati dell'esecuzione del flusso di lavoro secondario selezionando Esporta risultati dello stato della mappa su Amazon S3. Quindi, fornisci il nome del bucket Amazon S3 e il prefisso in cui desideri esportare i risultati.

Step Functions necessita delle autorizzazioni appropriate per accedere al bucket e alla cartella in cui si desidera esportare i risultati. Per informazioni sulla policy IAM richiesta, consulta. Politiche IAM per ResultWriter

Se esporti i risultati dell'esecuzione del flusso di lavoro secondario, l'esecuzione dello stato della mappa distribuita restituisce l'ARN Map Run e i dati sulla posizione di esportazione di Amazon S3 nel seguente formato:

```
{
  "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 esporta le esecuzioni con lo stesso stato nei rispettivi file. Ad esempio, se le esecuzioni dei flussi di lavoro secondari hanno prodotto 500 risultati di successo e 200 di errore, Step Functions crea due file nella posizione Amazon S3 specificata per i risultati di successo e di errore. In questo esempio, il file dei risultati di successo contiene i 500 risultati di successo, mentre il file dei risultati di errore contiene i 200 risultati di errore.

Per un determinato tentativo di esecuzione, Step Functions crea i seguenti file nella posizione Amazon S3 specificata in base all'output di esecuzione:

- `manifest.json`— Contiene metadati Map Run, come la posizione di esportazione, l'ARN di Map Run e informazioni sui file dei risultati.

  Se si utilizza redrivenun Map Run, il `manifest.json` file contiene riferimenti a tutte le esecuzioni di workflow secondarie riuscite in tutti i tentativi di Map Run. Tuttavia, questo file contiene riferimenti alle esecuzioni fallite e in sospeso per uno specifico caso. redrive
- `SUCCEEDED_n.json`— Contiene i dati consolidati per tutte le esecuzioni riuscite di workflow secondari. n rappresenta il numero di indice del file. Il numero di indice inizia da 0. Ad esempio, `SUCCEEDED_1.json`.
- `FAILED_n.json`— Contiene i dati consolidati per tutte le esecuzioni di workflow secondarie non riuscite, scadute e interrotte. Utilizzate questo file per eseguire il ripristino in caso di esecuzioni non riuscite. n rappresenta l'indice del file. Il numero di indice inizia da 0. Ad esempio, `FAILED_1.json`.
- `PENDING_n.json`— Contiene i dati consolidati per tutte le esecuzioni di workflow secondarie che non sono state avviate perché l'esecuzione della mappa non è riuscita o è stata interrotta. n rappresenta l'indice del file. Il numero di indice inizia da 0. Ad esempio, `PENDING_1.json`.

Step Functions supporta file di risultati individuali fino a 5 GB. Se la dimensione di un file supera i 5 GB, Step Functions crea un altro file per scrivere i risultati di esecuzione rimanenti e aggiunge un numero di indice al nome del file. Ad esempio, se la dimensione del `Succeeded_0.json` file supera i 5 GB, Step Functions crea un `Succeeded_1.json` file per registrare i risultati rimanenti.

Se non avete specificato di esportare i risultati dell'esecuzione del flusso di lavoro secondario, l'esecuzione della macchina a stati restituisce una matrice di risultati di esecuzione del flusso di lavoro secondario, come illustrato nell'esempio seguente:

> **Note**
>
> Se la dimensione di output restituita supera i 256 KB, l'esecuzione della macchina a stati fallisce e restituisce un `States.DataLimitExceeded` errore.

```
[
  {
    "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"
    }
  },
  ...
]
```

Politiche IAM per ResultWriter

Quando crei flussi di lavoro con la console Step Functions, Step Functions può generare automaticamente policy IAM basate sulle risorse nella definizione del flusso di lavoro. Queste politiche includono i privilegi minimi necessari per consentire al ruolo della macchina a stati di richiamare l'azione dell'`StartExecution`API per lo stato della mappa distribuita. Queste policy includono anche i privilegi minimi necessari Step Functions per accedere alle AWS risorse, come i bucket e gli oggetti Amazon S3 e le funzioni Lambda. Ti consigliamo vivamente di includere solo le autorizzazioni necessarie nelle tue policy IAM. Ad esempio, se il tuo flusso di lavoro include uno `Map` stato in modalità Distribuita, limita le policy allo specifico bucket e alla cartella Amazon S3 che contiene il set di dati.

> **Important**
>
> Se specifichi un bucket Amazon S3 e un oggetto, o prefisso, con un percorso di riferimento a una coppia chiave-valore esistente nell'input dello stato della mappa distribuita, assicurati di aggiornare le policy IAM per il tuo flusso di lavoro. Definisci le policy fino ai nomi dei bucket e degli oggetti in cui il percorso si risolve in fase di esecuzione.

Il seguente esempio di policy IAM concede i privilegi minimi richiesti per scrivere i risultati dell'esecuzione del flusso di lavoro secondario in una cartella denominata *`CSVJobs in un bucket`* Amazon S3 utilizzando l'azione API. `PutObject`

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
            "Resource": [
                "arn:aws:s3:::resultBucket/csvJobs/*"
            ]
        }
    ]
}
```

Se il bucket Amazon S3 su cui stai scrivendo il risultato dell'esecuzione del workflow secondario è crittografato utilizzando una AWS Key Management Service (AWS KMS) chiave, devi includere AWS KMS le autorizzazioni necessarie nella tua policy IAM. Per ulteriori informazioni, consulta Autorizzazioni IAM per bucket Amazon S3 AWS KMS key crittografato.

Analisi di un file CSV di input

Poiché non esiste un formato standardizzato per creare e gestire i dati nei file CSV, Step Functions analizza i file CSV in base alle seguenti regole:

- Le virgole (,) sono un delimitatore che separa i singoli campi.
- Le newline sono un delimitatore che separa i singoli record.
- I campi vengono trattati come stringhe. Per le conversioni dei tipi di dati, usa`States.StringToJson`funzione intrinseca in`ItemSelector`.
- Le virgolette doppie (» «) non sono necessarie per racchiudere le stringhe. Tuttavia, le stringhe racchiuse tra virgolette doppie possono contenere virgole e nuove righe senza che funzionino come delimitatori.
- Evita le virgolette doppie ripetendole.
- Se il numero di campi in una riga è inferiore al numero di campi nell'intestazione, Step Functions fornisce stringhe vuote per i valori mancanti.

- Se il numero di campi in una riga è superiore al numero di campi nell'intestazione, Step Functions salta i campi aggiuntivi.

Example dell'analisi di un file CSV di input

Supponiamo che tu abbia fornito un file CSV denominato*myCSVInput.csv*che contiene una riga come input. Quindi, hai memorizzati questo file in un bucket Amazon S3 che si chiama*my-bucket*. Il file CSV è il seguente.

```
abc,123,"This string contains commas, a double quotation marks (""), and a newline (
)",{""MyKey"":""MyValue""},"[1,2,3]"
```

La seguente macchina a stati legge questo file CSV e utilizzaItemSelectorper convertire i tipi di dati di alcuni campi.

```
{
  "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)"
      }
    }
  }
}
```

Quando si esegue questa macchina a stati, produce il seguente output.

```
[
  {
    "MyNumbers": 123,
    "MyObject": {
      "MyKey": "MyValue"
    },
    "MyString": "This string contains commas, a double quote (\"), and a newline (\n)",
    "MyLetters": "abc",
    "MyArray": [
      1,
      2,
      3
    ]
  }
]
```

Oggetto Context

L'oggetto context è una struttura JSON interna disponibile durante l'esecuzione e contiene informazioni sulla macchina a stati e sull'esecuzione. In questo modo i flussi di lavoro hanno accesso alle informazioni sulla loro esecuzione specifica. È possibile accedere all'oggetto contestuale dai seguenti campi:

- `InputPath`
- `OutputPath`
- `ItemsPath`(negli stati della mappa)
- `Variable`(negli stati Choice)
- `ResultSelector`
- `Parameters`
- Operatori di confronto da variabile a variabile

Formato oggetto contestuale

L'oggetto contestuale include informazioni sulla macchina a stati, lo stato, l'esecuzione e l'attività. Questo oggetto JSON include nodi per ogni tipo di dati ed è nel formato seguente.

```
{
    "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"
    }
}
```

Durante l'esecuzione, l'oggetto contestuale viene popolato con i dati rilevanti per il campo `Parameters` da cui viene effettuato l'accesso. Il valore per un campo `Task` è nullo se il campo `Parameters` è all'esterno dello stato di un'operazione.

Il valore dell'oggetto `RedriveCount` contestuale è 0, se non hai ancora eseguito redrivenun'esecuzione. Inoltre, l'oggetto `RedriveTime` contestuale è disponibile solo se hai redriven un'esecuzione. In caso redriven a Map Runaffermativo, l'oggetto `RedriveTime` contestuale è disponibile solo per i flussi di lavoro secondari di tipo Standard. Per un redriven Map Run con flussi di lavoro secondari di tipo Express, non `RedriveTime` è disponibile.

I contenuti di un'esecuzione in corso includono specifiche nel formato seguente.

```
{
    "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**
>
> Per i dati dell'oggetto contesto correlati agli stati `Map`, consulta Dati dell'oggetto contesto per gli stati Map.

Accesso all'oggetto contestuale

Per accedere all'oggetto contestuale, per prima cosa specificare il nome del parametro aggiungendo `.$` alla fine, come durante la selezione dell'input di stato con un percorso. Quindi, per accedere ai dati dell'oggetto contestuale dell'input, anteporre il percorso `$$.`. Questo indica AWS Step Functions di usare il percorso per selezionare un nodo nell'oggetto di contesto.

Gli esempi seguenti mostrano come accedere agli oggetti di contesto, come l'ID di esecuzione, il nome, l'ora di inizio e il redrive conteggio.

- Recupera e passa l'ARN di esecuzione a un servizio downstream
- Accedi all'ora e al nome di inizio dell'esecuzione in uno stato Pass
- Accedi al redrive conteggio di un'esecuzione

Recupera e passa l'ARN di esecuzione a un servizio downstream

Questo esempio di stato di attività utilizza un percorso per recuperare e passare l'esecuzione Amazon Resource Name (ARN) a un messaggio Amazon Simple Queue Service (Amazon SQS).

```
{
  "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"
  }
```

```
}
```

Per ulteriori informazioni su come usare il token di attività durante la chiamata a un servizio integrato, consultare Attendere un callback con il token dell'attività.

Accedi all'ora e al nome di inizio dell'esecuzione in uno stato 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
    }
  }
}
```

Accedi al redrive conteggio di un'esecuzione

Il seguente esempio di definizione dello stato di un'attività mostra come accedere al redriveconteggio di un'esecuzione.

```
{
  "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
}
```

Dati dell'oggetto contesto per gli stati Map

Sono disponibili due elementi aggiuntivi nell'oggetto contesto durante l'elaborazione di uno stato stato Map: `Index` e `Value`. Per ogni iterazione di `Map` stato, `Index` contiene il numero di indice dell'elemento dell'array attualmente in fase di elaborazione, mentre `Value` contiene l'elemento dell'array in fase di elaborazione. All'interno di uno `Map` stato, l'oggetto contesto include i seguenti dati:

```
"Map": {
   "Item": {
      "Index": Number,
      "Value": "String"
   }
}
```

Questi sono disponibili solo in uno `Map` stato e possono essere specificati nel `ItemSelector` campo.

> **Note**
>
> È necessario definire i parametri dall'oggetto contesto nel blocco `ItemSelector` dello stato `Map` principale, non all'interno degli stati inclusi nella sezione `ItemProcessor`.

Data una macchina a stati con uno stato `Map` semplice, possiamo inserire informazioni dall'oggetto contesto come segue.

```
{
  "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
    }
  }
}
```

Se esegui la macchina a stati precedente con il seguente input, `Index` e `Value` vengono inseriti nell'output.

```
[
  {
    "who": "bob"
  },
  {
    "who": "meg"
  },
  {
    "who": "joe"
  }
]
```

L'output per l'esecuzione restituisce i valori `Index` e `Value` gli elementi per ciascuna delle tre iterazioni nel modo seguente:

```
[
  {
    "ContextIndex": 0,
    "ContextValue": {
      "who": "bob"
    }
  },
  {
    "ContextIndex": 1,
    "ContextValue": {
      "who": "meg"
    }
```

```
  },
  {

    "ContextIndex": 2,
    "ContextValue": {
      "who": "joe"
    }
  }
]
```

Simulatore di flusso di dati

Puoi progettare, implementare ed eseguire il debug dei flussi di lavoro nella console Step Functions. Puoi anche controllare il flusso di dati nei tuoi flussi di lavoro con l'elaborazione dei dati JsonPathdi input e output. Con il simulatore di flusso di dati, puoi simulare l'ordine in cui Attività gli stati del tuo flusso di lavoro elaborano i dati in fase di esecuzione. Utilizzando il simulatore, puoi capire come filtrare e manipolare i dati mentre fluiscono da uno stato all'altro. Simula ciascuno dei seguenti campi utilizzati da Step Functions per elaborare e controllare il flusso di dati JSON:

`InputPath`

Seleziona QUALE parte dell'intero payload di input da utilizzare come input di un'attività. Se si specifica questo campo, Step Functions applica innanzitutto questo campo.

`Parametri`

Specifica COME deve apparire l'input prima di richiamare l'operazione. Con il `Parameters` campo, puoi creare una raccolta di coppie chiave-valore che vengono passate come input a un'Servizio AWSintegrazione, ad esempio una AWS Lambda funzione. Questi valori possono essere statici o selezionati dinamicamente dall'input di stato o dall'oggetto di contesto del flusso di lavoro.

`ResultSelector`

Determina COSA scegliere dall'output di un'attività. Con il `ResultSelector` campo, puoi creare una raccolta di coppie chiave-valore che sostituiscono il risultato di uno stato e passano quella raccolta a. `ResultPath`

`ResultPath`

Determina DOVE inserire l'output di un'attività. Utilizzate il `ResultPath` per determinare se l'output di uno stato è una copia del suo input, il risultato che produce o una combinazione di entrambi.

`OutputPath`

Determina COSA inviare allo stato successivo. Con`OutputPath`, puoi filtrare le informazioni indesiderate e trasmettere solo la parte di dati JSON che ti interessa.

In questo argomento

- Utilizzo del simulatore di flusso di dati
- Considerazioni sull'utilizzo del simulatore di flusso di dati

Utilizzo del simulatore di flusso di dati

Il simulatore fornisce un side-by-side confronto in tempo reale dei dati prima e dopo l'applicazione di un campo di elaborazione dei dati di input e output. Per utilizzare il simulatore, specificate un input JSON. Quindi, valutalo attraverso ciascuno dei campi di elaborazione di input e output. Il simulatore convalida automaticamente l'input JSON ed evidenzia eventuali errori di sintassi.

Per utilizzare il simulatore di flusso di dati

Nei passaggi seguenti, fornisci l'input JSON e applichi i `Parametri` campi `InputPath` and. Puoi anche applicare gli altri campi disponibili e visualizzarne i risultati.

1. Apri la console Step Functions.
2. Nel riquadro di navigazione, scegli Simulatore di flusso di dati.
3. Nell'area di immissione dello stato, sostituisci i dati JSON di esempio precompilati con i seguenti dati JSON. Quindi, seleziona Next (Successivo).

```
{
  "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. Per InputPath, inserisci **`$.data.address`** per selezionare il nodo di indirizzo dei dati JSON di input.

   La InputPath casella Stato input after mostra i seguenti risultati.

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

5. Seleziona Successivo.
6. Applica il `Parameters` campo per convertire i dati JSON risultanti in una stringa. Per convertire i dati, procedi come segue:

   - Nella casella Parametri, inserisci il codice seguente per creare una stringa chiamata`addressString`.

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

7. Visualizza il risultato dell'applicazione sul `Parameters` campo nella casella Input filtrato dopo i parametri.

Considerazioni sull'utilizzo del simulatore di flusso di dati

Prima di utilizzare il simulatore di flusso di dati, considera i suoi limiti, tra cui, a titolo esemplificativo:

- Espressioni di filtro non supportate

  Le espressioni di filtro nel simulatore si comportano in modo diverso rispetto al servizio Step Functions. Ciò include espressioni che utilizzano la seguente sintassi:`[?(expression)]`. Di seguito è riportato un elenco di espressioni non supportate. Se utilizzate, queste espressioni potrebbero non restituire il risultato previsto dopo la loro valutazione.

  - `$..book[?(@.isInStock==true)]`
  - `$..book[?(@.price > 10.0)].title`

- JsonPathValutazione errata per gli array a singolo elemento

  Se filtri i tuoi dati con un'JsonPathespressione che restituirebbe un singolo elemento dell'array, il simulatore restituisce l'elemento senza l'array. Ad esempio, considera la seguente matrice di dati, chiamata`items`:

```
{
  "items": [
    {
      "name": "shoe",
      "color": "blue",
      "comment": "nice shoe"
    },
    {
      "name": "hat",
      "color": "red"
    },
    {
      "name": "shirt",
      "color": "yellow"
    }
  ]
}
```

  Dato questo `items` array, se inserisci il **`$..comment`** InputPath campo, ti aspetteresti il seguente risultato:

```
[
  "nice shoe"
]
```

Tuttavia, il simulatore di flusso di dati restituisce invece il seguente output:

```
"nice shoe"
```

Per la JsonPath valutazione di un array che contiene più elementi, il simulatore restituisce l'output previsto.

Gestisci le distribuzioni continue con versioni e alias

Puoi utilizzare Step Functions per gestire le distribuzioni continue dei tuoi flussi di lavoro tramite versioni e alias delle macchine a stato. Una versione è un'istantanea numerata e immutabile di una macchina a stati che è possibile eseguire. Un alias è un puntatore per un massimo di due versioni di una macchina a stati.

Puoi gestire più versioni delle tue macchine statali e gestirne l'implementazione nel flusso di lavoro di produzione. Con gli alias, puoi indirizzare il traffico tra diverse versioni del flusso di lavoro e distribuire gradualmente tali flussi di lavoro nell'ambiente di produzione.

Inoltre, è possibile avviare le esecuzioni delle macchine a stati utilizzando una versione o un alias. Se non si utilizza una versione o un alias quando si avvia l'esecuzione di una macchina a stati, Step Functions utilizza la revisione più recente della definizione della macchina a stati.

> **Revisione della macchina statale**
>
> Una macchina a stati può avere una o più revisioni. Quando si aggiorna una macchina a stati utilizzando l'azione UpdateStateMachineAPI, viene creata una nuova revisione della macchina a stati. Una revisione è un'istantanea immutabile e di sola lettura della definizione e della configurazione di una macchina a stati. Non è possibile avviare l'esecuzione di una macchina a stati da una revisione e le revisioni non hanno un ARN. Le revisioni hanno un`revisionId`, che è un identificatore univoco universale (UUID).

Indice

- Versioni di macchine a stati
- Alias di macchine a stati
- Autorizzazione per versioni e alias

- Associazione di esecuzioni di macchine a stati con una versione o un alias
- Esempio di distribuzione di alias e versioni
- Esegui la distribuzione graduale delle versioni di State Machine

Versioni di macchine a stati

Una versione è un'istantanea numerata e immutabile di una macchina a stati. Si pubblicano versioni a partire dalla revisione più recente apportata a quella macchina a stati. Ogni versione ha un Amazon Resource Name (ARN) univoco. Questo ARN è una combinazione dell'ARN della macchina a stati e del numero di versione separati da due punti (:). L'esempio seguente mostra il formato della versione ARN di una macchina a stati.

```
arn:partition:states:region:account-id:stateMachine:myStateMachine:1
```

Per iniziare a utilizzare le versioni delle macchine a stati, è necessario pubblicare la prima versione. Dopo aver pubblicato una versione, puoi richiamare l'azione StartExecutionAPI con la versione ARN. Non puoi modificare una versione, ma puoi aggiornare una macchina a stati e pubblicare una nuova versione. Puoi anche pubblicare più versioni della tua macchina a stati.

Quando pubblicate una nuova versione della vostra macchina a stati, Step Functions le assegna un numero di versione. I numeri di versione iniziano da 1 e aumentano in modo monotono per ogni nuova versione. I numeri di versione non vengono riutilizzati per una determinata macchina a stati. Se elimini la versione 10 della tua macchina a stati e poi pubblichi una nuova versione, Step Functions la pubblica come versione 11.

Le seguenti proprietà sono le stesse per tutte le versioni di una macchina a stati:

- Tutte le versioni di una macchina a stati condividono lo stesso tipo (Standard o Express).
- Non è possibile modificare il nome o la data di creazione di una macchina a stati tra le versioni.

- I tag si applicano a livello globale alle macchine a stati. È possibile gestire i tag per le macchine a stati utilizzando le azioni TagResourcee UntagResourceAPI.

Le macchine a stati contengono anche proprietà che fanno parte di ogni versionerevision, ma queste proprietà possono differire tra due versioni o revisioni specifiche. Queste proprietà includono la definizione della macchina a stati, il ruolo IAM, la configurazione di tracciamento e la configurazione di registrazione.

Indice

- Pubblicazione di una versione di una macchina a stati (Console)
- Gestione delle versioni con le operazioni dell'API Step Functions
- Esecuzione di una versione di macchina a stati dalla console

Pubblicazione di una versione di una macchina a stati (Console)

È possibile pubblicare fino a 1000 versioni di una macchina a stati. Per richiedere un aumento di questo limite flessibile, utilizza la pagina Support Center nel AWS Management Console. È possibile eliminare manualmente le versioni non utilizzate dalla console o richiamando l'azione DeleteStateMachineVersionAPI.

Per pubblicare una versione di una macchina a stati

1. Apri la console Step Functions, quindi scegli una macchina a stati esistente.
2. Nella pagina dei dettagli della macchina a stati, scegliete Modifica.
3. Modifica la definizione della macchina a stati come richiesto, quindi scegli Salva.
4. Scegliere Publish version (Pubblica versione).
5. (Facoltativo) Nel campo Descrizione della finestra di dialogo che appare, inserisci una breve descrizione sulla versione della macchina a stati.
6. Seleziona Publish (Pubblica).

> **Note**
>
> Quando pubblicate una nuova versione della vostra macchina a stati, Step Functions le assegna un numero di versione. I numeri di versione iniziano da 1 e aumentano in modo monotono per ogni nuova versione. I numeri di versione non vengono riutilizzati per una

determinata macchina a stati. Se elimini la versione 10 della tua macchina a stati e poi pubblichi una nuova versione, Step Functions la pubblica come versione 11.

Gestione delle versioni con le operazioni dell'API Step Functions

Step Functions fornisce le seguenti operazioni API per pubblicare e gestire le versioni delle macchine a stati:

- PublishStateMachineVersion— Pubblica una versione della versione corrente revision di una macchina a stati.
- UpdateStateMachine— Pubblica una nuova versione della macchina a stati se si aggiorna una macchina a stati e si imposta il `publish` parametro su `true` nella stessa richiesta.
- CreateStateMachine— Pubblica la prima revisione della macchina a stati se si imposta il `publish` parametro su. `true`
- ListStateMachineVersions— Elenca le versioni per l'ARN della macchina a stati specificata.
- DescribeStateMachine— Restituisce i dettagli della versione della macchina a stati per una versione ARN specificata in. `stateMachineArn`
- DeleteStateMachineVersion— Elimina la versione di una macchina a stati.

Per pubblicare una nuova versione della revisione corrente di una macchina a stati chiamata *`myStateMachine`* utilizzando il AWS Command Line Interface, utilizzare il `publish-state-machine-version` comando:

```
aws stepfunctions publish-state-machine-version --state-machine-arn arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine
```

La risposta restituisce il `stateMachineVersionArn`. Ad esempio, il comando precedente restituisce una risposta di`arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:1`.

> **Note**
>
> Quando pubblicate una nuova versione della vostra macchina a stati, Step Functions le assegna un numero di versione. I numeri di versione iniziano da 1 e aumentano in modo monotono per ogni nuova versione. I numeri di versione non vengono riutilizzati per una

determinata macchina a stati. Se elimini la versione 10 della tua macchina a stati e poi pubblichi una nuova versione, Step Functions la pubblica come versione 11.

Esecuzione di una versione di macchina a stati dalla console

Per iniziare a utilizzare le versioni delle macchine a stati, è necessario prima pubblicare una versione dalla macchina a stati correnterevision. Per pubblicare una versione, usa la console Step Functions o richiama l'azione PublishStateMachineVersionAPI. Puoi anche richiamare l'azione UpdateStateMachineAliasAPI con un parametro opzionale denominato `publish` per aggiornare una macchina a stati e pubblicarne la versione.

È possibile avviare l'esecuzione di una versione utilizzando la console o richiamando l'azione StartExecutionAPI e fornendo la versione ARN. Puoi anche usare un alias per avviare le esecuzioni di una versione. In base alla configurazione di routing, un alias indirizza il traffico verso una versione specifica.

Se si avvia l'esecuzione di una macchina a stati senza utilizzare una versione, Step Functions utilizza la revisione più recente della macchina a stati per l'esecuzione. Per informazioni su come Step Functions associa un'esecuzione a una versione, vedereAssociazione di esecuzioni a una versione o a un alias.

Per avviare un'esecuzione utilizzando una versione di macchina a stati

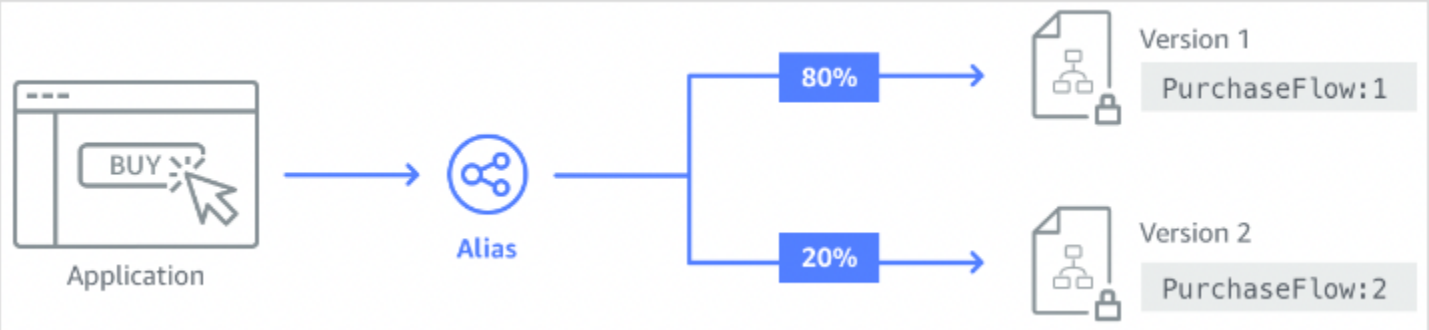
1. Apri la console Step Functions, quindi scegli una macchina a stati esistente per la quale hai pubblicato una o più versioni. Per informazioni su come pubblicare una versione, consultaPubblicazione di una versione di una macchina a stati (Console).
2. Nella pagina dei dettagli della macchina a stati, scegli la scheda Versioni.
3. Nella sezione Versioni, procedi come segue:
   a. Seleziona la versione con cui vuoi iniziare l'esecuzione.
   b. Selezionare Start execution (Avvia esecuzione).
4. (Facoltativo) Nella finestra di dialogo Avvia esecuzione, inserite un nome per l'esecuzione.
5. (Facoltativo), inserite l'input di esecuzione, quindi scegliete Avvia esecuzione.

Alias di macchine a stati

Un alias è un puntatore per un massimo di due versioni della stessa macchina a stati. È possibile creare più alias per le proprie macchine a stati. Ogni alias ha un Amazon Resource Name (ARN) univoco. L'alias ARN è una combinazione dell'ARN della macchina a stati e del nome dell'alias, separati da due punti (:). L'esempio seguente mostra il formato di un alias ARN di una macchina a stati.

```
arn:partition:states:region:account-id:stateMachine:myStateMachine:aliasName
```

È possibile utilizzare un alias per indirizzare il traffico tra una delle due versioni della macchina a stati. È inoltre possibile creare un alias che punti a una singola versione. Gli alias possono puntare solo a versioni di macchine a stati. Non è possibile utilizzare un alias per puntare a un altro alias. È inoltre possibile aggiornare un alias in modo che punti a una versione diversa della macchina a stati.



Indice

- Creazione di un alias per una macchina a stati (Console)
- Gestione degli alias con le operazioni dell'API Step Functions
- Configurazione del routing dell'alias
- Esecuzione di una macchina a stati utilizzando un alias (Console)

Creazione di un alias per una macchina a stati (Console)

È possibile creare fino a 100 alias per ogni macchina a stati utilizzando la console Step Functions o richiamando l'azione CreateStateMachineAliasAPI. Per richiedere un aumento di questo limite flessibile, utilizza la pagina Support Center nel AWS Management Console. Elimina gli alias non utilizzati dalla console o richiamando l'DeleteStateMachineAliasazione API.

Per creare un alias di una macchina a stati

1. Apri la console Step Functions, quindi scegli una macchina a stati esistente.
2. Nella pagina dei dettagli della macchina a stati, scegliete la scheda Alias.
3. Scegli Crea nuovo alias.
4. Nella pagina Create alias (Crea alias), eseguire le operazioni seguenti:
   a. Inserisci un nome alias.
   b. (Facoltativo) Immettere una Description (Descrizione) per l'alias.
5. Per configurare il routing sull'alias, vedi Configurazione del routing degli alias.
6. Scegli Crea alias.

Gestione degli alias con le operazioni dell'API Step Functions

Step Functions fornisce le seguenti operazioni API che è possibile utilizzare per creare e gestire alias di macchine a stati o ottenere informazioni sugli alias:

- CreateStateMachineAlias— Crea un alias per una macchina a stati.
- DescribeStateMachineAlias— Restituisce dettagli su un alias di una macchina a stati.
- ListStateMachineAliases— Elenca gli alias per l'ARN della macchina a stati specificata.
- UpdateStateMachineAlias— Aggiorna la configurazione di un alias di una macchina a stati esistente modificandone o. `description` `routingConfiguration`
- DeleteStateMachineAlias— Elimina la versione di una macchina a stati.

Per creare un alias denominato *`PROD`* che punti alla versione 1 di una macchina a stati denominata *`myStateMachine`* utilizzando il AWS Command Line Interface, utilizzare il `create-state-machine-alias` comando.

```
aws stepfunctions create-state-machine-alias --name PROD  --routing-
configuration "[{\"stateMachineVersionArn\":\"arn:aws:states:us-
east-1:123456789012:stateMachine:myStateMachine:1\",\"weight\":100}]"
```

Configurazione del routing dell'alias

È possibile utilizzare un alias per indirizzare il traffico di esecuzione tra due versioni di una macchina a stati. Ad esempio, supponiamo che tu voglia lanciare una nuova versione della tua macchina a

stati. È possibile ridurre i rischi associati alla distribuzione della nuova versione configurando il routing su un alias. Configurando il routing, è possibile inviare la maggior parte del traffico a una versione precedente e testata della macchina a stati. La nuova versione può quindi ricevere una percentuale inferiore, fino a quando non confermerai che il rollforward della nuova versione è sicuro.

Per definire la configurazione del routing, assicurati di pubblicare entrambe le versioni della macchina a stati a cui fa riferimento il tuo alias. Quando si avvia un'esecuzione da un alias, Step Functions sceglie in modo casuale la versione della macchina a stati da eseguire tra le versioni specificate nella configurazione del routing. Basa questa scelta sulla percentuale di traffico assegnata a ciascuna versione nella configurazione di routing degli alias.

Per configurare la configurazione del routing su un alias

- Nella pagina Crea alias, in Configurazione del routing, procedi come segue:

  a. Per Versione, scegli la prima versione della macchina a stati a cui punta l'alias.

  b. Seleziona la casella di controllo Dividi il traffico tra due versioni.

  > **Tip**
  >
  > Per puntare a una singola versione, deseleziona la casella di controllo Dividi il traffico tra due versioni.

  c. Per Versione, scegli la seconda versione a cui deve puntare l'alias.

  d. Nei campi Percentuale di traffico, specifica la percentuale di traffico da indirizzare verso ciascuna versione. Ad esempio, inserisci **`60`** e **`40`** indirizza il 60 percento del traffico di esecuzione verso la prima versione e il 40 percento del traffico verso la seconda versione.

     Le percentuali di traffico combinate devono essere pari al 100 percento.

Esecuzione di una macchina a stati utilizzando un alias (Console)

È possibile avviare le esecuzioni delle macchine a stati con un alias dalla console o richiamando l'azione StartExecutionAPI con l'ARN dell'alias. Step Functions esegue quindi la versione specificata dall'alias. Per impostazione predefinita, se non si specifica una versione o un alias quando si avvia l'esecuzione di una macchina a stati, Step Functions utilizza la revisione più recente.

Per avviare l'esecuzione di una macchina a stati utilizzando un alias

1. Apri la console Step Functions, quindi scegli una macchina a stati esistente per la quale hai creato un alias. Per informazioni sulla creazione di un alias, consulta. Creazione di un alias per una macchina a stati (Console)
2. Nella pagina dei dettagli della macchina a stati, scegli la scheda Alias.
3. Nella sezione Alias, procedi come segue:
   a. Seleziona l'alias con cui vuoi iniziare l'esecuzione.
   b. Selezionare Start execution (Avvia esecuzione).
4. (Facoltativo) Nella finestra di dialogo Avvia esecuzione, immettete un nome per l'esecuzione.
5. Se necessario, immettete l'input di esecuzione, quindi scegliete Avvia esecuzione.

Autorizzazione per versioni e alias

Per richiamare le azioni API di Step Functions con una versione o un alias, sono necessarie le autorizzazioni appropriate. Per autorizzare una versione o un alias a richiamare un'azione API, Step Functions utilizza l'ARN della macchina a stati anziché utilizzare la versione ARN o l'alias ARN. Puoi anche limitare le autorizzazioni per una versione o un alias specifici. Per ulteriori informazioni, consulta Ridurre l'ambito delle autorizzazioni.

È possibile utilizzare il seguente esempio di policy IAM di una macchina a stati denominata *myStateMachine* per richiamare l'azione CreateStateMachineAliasAPI per creare un alias di macchina a stati.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "states:CreateStateMachineAlias",
      "Resource": "arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine"
    }
  ]
}
```

Quando imposti le autorizzazioni per consentire o negare l'accesso alle azioni API utilizzando versioni o alias della macchina a stati, considera quanto segue:

- Se si utilizza il `publish` parametro delle azioni CreateStateMachinee UpdateStateMachineAPI per pubblicare una nuova versione della macchina a stati, è necessaria anche l'`ALLOW`autorizzazione sull'azione PublishStateMachineVersionAPI.
- L'azione DeleteStateMachineAPI elimina tutte le versioni e gli alias associati a una macchina a stati.

In questo argomento

- Definizione delle autorizzazioni per una versione o un alias

Definizione delle autorizzazioni per una versione o un alias

È possibile utilizzare un qualificatore per limitare ulteriormente l'autorizzazione necessaria a una versione o a un alias. Un qualificatore si riferisce a un numero di versione o a un nome alias. Il qualificatore viene utilizzato per qualificare una macchina statale. L'esempio seguente è un ARN di una macchina a stati che utilizza un alias denominato `PROD` come qualificatore.

```
arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:PROD
```

Per ulteriori informazioni sugli ARN qualificati e non qualificati, vedere. Associazione di esecuzioni a una versione o a un alias

Puoi definire le autorizzazioni utilizzando la chiave di contesto opzionale indicata `states:StateMachineQualifier` nella `Condition` dichiarazione di una policy IAM. Ad esempio, la seguente politica IAM per una macchina a stati denominata `myStateMachine` nega l'accesso per richiamare l'azione DescribeStateMachineAPI con un alias denominato come `PROD` o versione. 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"
          ]
        }
      }
    }
  ]
}
```

L'elenco seguente specifica le azioni API su cui è possibile definire le autorizzazioni con la chiave di `StateMachineQualifier` contesto.

- CreateStateMachineAlias
- DeleteStateMachineAlias
- DeleteStateMachineVersion
- DescribeStateMachine
- DescribeStateMachineAlias
- ListExecutions
- ListStateMachineAliases
- StartExecution
- StartSyncExecution
- UpdateStateMachineAlias

Associazione di esecuzioni di macchine a stati con una versione o un alias

Step Functions associa un'esecuzione a una versione o alias basato sull'Amazon Resource Name (ARN) che usi per richiamare l'azione API. StartExecution Step Functions esegue questa azione all'inizio dell'esecuzione.

È possibile avviare l'esecuzione di una macchina a stati utilizzando un ARN qualificato o non qualificato.

- ARN qualificato: si riferisce a un ARN di una macchina a stati con il suffisso un numero di versione o un nome alias.

  Il seguente esempio di ARN qualificato si riferisce alla versione 3 di una macchina a stati denominata. `myStateMachine`

```
arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:3
```

Il seguente esempio di ARN qualificato si riferisce a un alias denominato `PROD` di una macchina a stati denominata. `myStateMachine`

```
arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:PROD
```

- ARN non qualificato: si riferisce all'ARN di una macchina a stati senza un numero di versione o un suffisso del nome alias.

```
arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine
```

Ad esempio, se l'ARN qualificato si riferisce alla versione`3`, Step Functions associa l'esecuzione a questa versione. Non associa l'esecuzione a nessun alias che rimanda alla versione. `3`

Se l'ARN qualificato fa riferimento a un alias, Step Functions associa l'esecuzione a quell'alias e alla versione a cui fa riferimento l'alias. Un'esecuzione può essere associata a un solo alias.

> **Note**
>
> Se si avvia un'esecuzione con un ARN non qualificato, Step Functions non associa tale esecuzione a una versione anche se la versione utilizza la stessa macchina a stati. revision Ad esempio, se la versione 3 utilizza la revisione più recente, ma si avvia un'esecuzione con un ARN non qualificato, Step Functions non associa tale esecuzione alla versione 3.

In questo argomento

- Visualizzazione delle esecuzioni iniziate con una versione o un alias

Visualizzazione delle esecuzioni iniziate con una versione o un alias

Step Functions offre i seguenti modi in cui è possibile visualizzare le esecuzioni iniziate con una versione o un alias:

Utilizzo delle azioni API

Puoi visualizzare tutte le esecuzioni associate a una versione o a un alias richiamando le azioni DescribeExecutione ListExecutionsAPI. Queste azioni API restituiscono l'ARN della versione o dell'alias utilizzato per avviare l'esecuzione. Queste azioni restituiscono anche altri dettagli, tra cui lo stato e l'ARN dell'esecuzione.

Puoi anche fornire un alias di macchina a stati ARN o versione ARN per elencare le esecuzioni associate a un alias o a una versione specifici.

Il seguente esempio di risposta dell'azione ListExecutionsAPI mostra l'ARN dell'alias utilizzato per avviare un'esecuzione di una macchina a stati denominata. *myFirstExecution*

Il testo in *corsivo* nel seguente frammento di codice rappresenta informazioni specifiche sulla risorsa.

```
{
    "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"
        }
    ]
}
```

Utilizzo della console Step Functions

Puoi anche visualizzare le esecuzioni avviate da una versione o da un alias dalla console Step Functions. La procedura seguente mostra come visualizzare le esecuzioni avviate con una versione specifica:

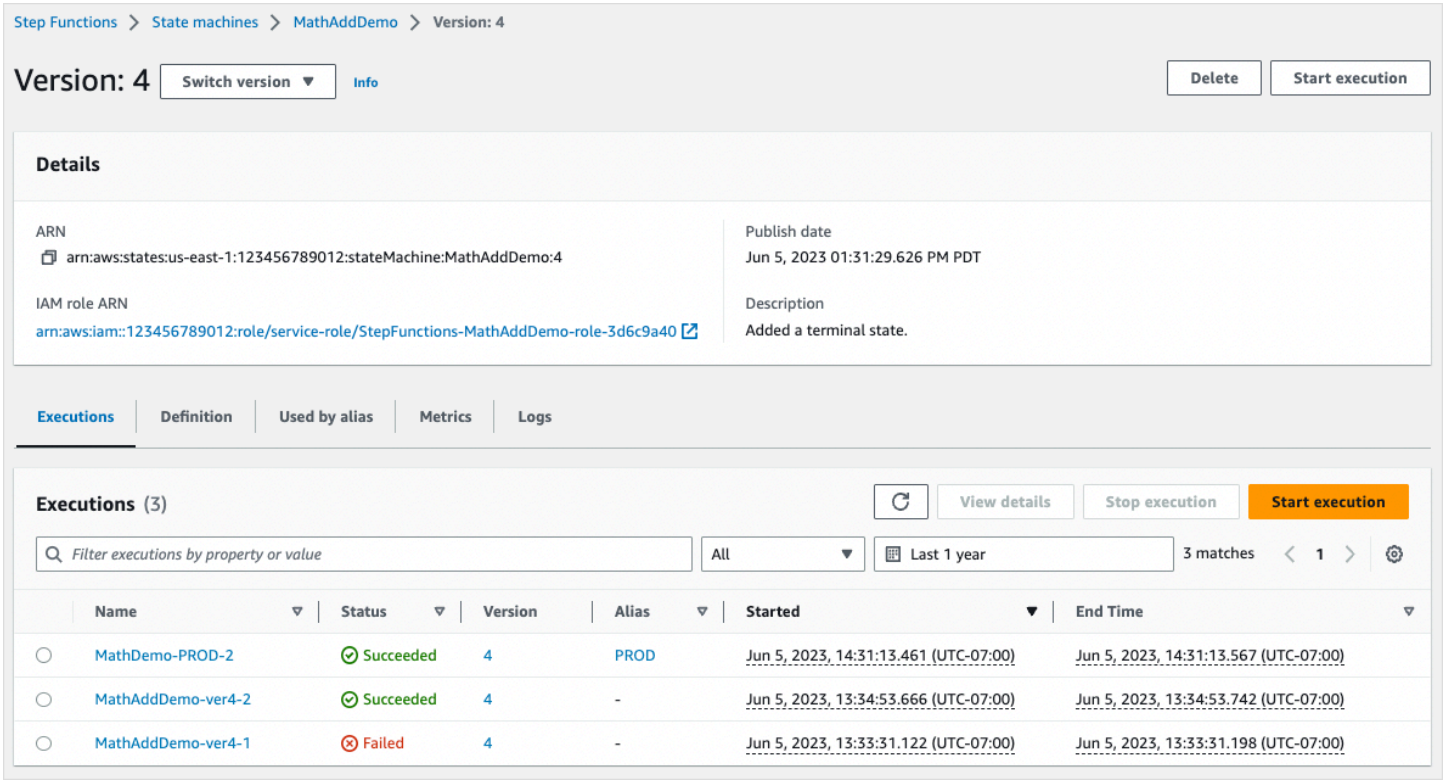
1. Apri la console Step Functions, quindi scegli una macchina a stati esistente per la quale hai pubblicato una versione o creato un alias. Questo esempio mostra come visualizzare le esecuzioni iniziate con una versione specifica della macchina a stati.

2. Scegliete la scheda Versioni, quindi scegliete una versione dall'elenco Versioni.

> **Tip**
>
> Filtra per proprietà o casella di valore per cercare una versione specifica.

3. Nella pagina dei dettagli della versione, è possibile visualizzare un elenco di tutte le esecuzioni in corso e passate di macchine a stati avviate con la versione selezionata.

L'immagine seguente mostra la pagina della console con i dettagli della versione. Questa pagina elenca le esecuzioni avviate dalla versione 4 di una macchina a stati denominata*MathAddDemo*. Questo elenco mostra anche un'esecuzione che è stata avviata da un alias denominato. *PROD* Questo alias ha indirizzato il traffico di esecuzione alla versione 4.



Utilizzo dei parametri CloudWatch

Per ogni esecuzione di una macchina a stati che si inizia con aQualified ARN, Step Functions emette metriche aggiuntive con lo stesso nome e valore delle metriche emesse attualmente. Queste metriche aggiuntive contengono dimensioni per ogni identificatore di versione e nome alias con cui si avvia un'esecuzione. Con queste metriche, è possibile monitorare le esecuzioni delle macchine a

stati a livello di versione e determinare quando potrebbe essere necessario uno scenario di rollback. Puoi anche creare CloudWatch allarmi Amazon in base a questi parametri.

Step Functions emette le seguenti metriche per le esecuzioni che iniziano con un alias o una versione:

- `ExecutionTime`
- `ExecutionsAborted`
- `ExecutionsFailed`
- `ExecutionsStarted`
- `ExecutionsSucceeded`
- `ExecutionsTimedOut`

Se hai avviato l'esecuzione con una versione ARN, Step Functions pubblica la metrica con `StateMachineArn` e una seconda metrica con e dimensioni. `StateMachineArn` `Version`

Se hai avviato l'esecuzione con un alias ARN, Step Functions emette le seguenti metriche:

- Due metriche per l'ARN e la versione non qualificati.
- Una metrica con le dimensioni e. `StateMachineArn` `Alias`

Esempio di distribuzione di alias e versioni

Il seguente esempio della tecnica di distribuzione di Canary mostra come è possibile implementare una nuova versione della macchina a stati con. AWS Command Line Interface In questo esempio, l'alias creato indirizza il 20% del traffico di esecuzione verso la nuova versione. Quindi indirizza il restante 80 percento della versione precedente. Per implementare una nuova versione della macchina a stati e spostare il traffico di esecuzione con un alias, completa i seguenti passaggi:

1. Pubblica una versione dalla revisione corrente della macchina a stati.

   Usa il publish-state-machine-version comando in AWS CLI per pubblicare una versione dalla revisione corrente di una macchina a stati chiamata *`myStateMachine`*:

   ```
   aws stepfunctions publish-state-machine-version --state-machine-arn
    arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine
   ```

La risposta restituisce `stateMachineVersionArn` la versione che hai pubblicato. Ad esempio, `arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:1`.

2. Crea un alias che punti alla versione della macchina a stati.

   Usa il create-state-machine-alias comando per creare un alias denominato *PROD* che punti alla versione 1 di*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. Verifica che le esecuzioni avviate dall'alias utilizzino la versione pubblicata corretta.

   Avvia una nuova esecuzione di *myStateMachine* fornendo l'ARN dell'alias **PROD** nel start-execution comando:

   ```
   aws stepfunctions start-execution
     --state-machine-arn arn:aws:states:us-
   east-1:123456789012:stateMachineAlias:myStateMachine:PROD
     --input "{}"
   ```

   Se fornisci l'ARN della macchina a stati nella StartExecutionrichiesta, questa utilizza la più recente revision della macchina a stati anziché la versione specificata nell'alias per avviare l'esecuzione.

4. Aggiorna la definizione della macchina a stati e pubblica una nuova versione.

   Aggiorna *myStateMachine* e pubblica la sua nuova versione. Per fare ciò, usa il `publish` parametro opzionale del update-state-machine comando:

   ```
   aws stepfunctions update-state-machine
       --state-machine-arn arn:aws:states:us-
   east-1:123456789012:stateMachine:myStateMachine
       --definition $UPDATED_STATE_MACHINE_DEFINITION
       --publish
   ```

   La risposta restituisce il valore `stateMachineVersionArn` per la nuova versione. Ad esempio, `arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:2`.

5. Aggiorna l'alias in modo che punti a entrambe le versioni e imposta la configurazione del routing dell'alias.

Usa il update-state-machine-alias comando per aggiornare la configurazione di routing dell'aliasPROD. Configura l'alias in modo che l'80% del traffico di esecuzione vada alla versione 1 e il restante 20% alla versione 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. Sostituisci la versione 1 con la versione 2.

   Dopo aver verificato che la nuova versione della macchina a stati funzioni correttamente, puoi distribuire la nuova versione della macchina a stati. A tale scopo, aggiorna nuovamente l'alias per assegnare il 100% del traffico di esecuzione alla nuova versione.

   Usa il update-state-machine-alias comando per impostare la configurazione di routing dell'alias PROD al 100% per la versione 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

   Per ripristinare la distribuzione della versione 2, modifica la configurazione di routing dell'alias per spostare il 100% del traffico verso la nuova versione implementata.

   ```
   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}]"
   ```

È possibile utilizzare versioni e alias per eseguire altri tipi di distribuzioni. Ad esempio, puoi eseguire una distribuzione continua di una nuova versione della tua macchina statale. A tal fine, aumenta gradualmente la percentuale ponderata nella configurazione di routing dell'alias che rimanda alla nuova versione.

Puoi anche usare versioni e alias per eseguire una distribuzione blu/verde. Per farlo, crea un alias denominato `green` che esegua la versione corrente 1 della tua macchina a stato. Quindi, crea un altro alias denominato `blue` che esegua la nuova versione, *2* ad esempio. Per testare la nuova versione, invia il traffico di esecuzione all'`blue`alias. Quando sei sicuro che la nuova versione funzioni correttamente, aggiorna l'`green`alias in modo che punti alla nuova versione.

Esegui la distribuzione graduale delle versioni di State Machine

Una distribuzione in successione è una strategia di distribuzione che sostituisce lentamente le versioni precedenti di un'applicazione con nuove versioni di un'applicazione. Per eseguire una distribuzione continua di una versione di una macchina a stato, invia gradualmente una quantità crescente di traffico di esecuzione alla nuova versione. La quantità di traffico e il tasso di aumento sono parametri che puoi configurare.

È possibile eseguire la distribuzione progressiva di una versione utilizzando una delle seguenti opzioni:

- Console Step Functions— Crea un alias che punti a due versioni della stessa macchina a stati. Per questo alias, configuri la configurazione del routing per spostare il traffico tra le due versioni. Per ulteriori informazioni sull'utilizzo della console per implementare le versioni, vedereVersioneAlias.
- Script perAWS CLIe SDK— Creare uno script di shell usandoAWS CLIo ilAWSSDK. Per ulteriori informazioni, consulta le seguenti sezioni sull'utilizzoAWS CLIeAWSSDK.
- AWS CloudFormationmodelli— Usa il`AWS::StepFunctions::StateMachineVersion`e`AWS::StepFunctions::StateMachineAlias`r per pubblicare più versioni di macchine a stati e creare un alias che punti a una o due di queste versioni.

Usa ilAWS CLIper implementare una nuova versione della macchina a stati

Lo script di esempio in questa sezione mostra come è possibile utilizzareAWS CLIper spostare gradualmente il traffico da una versione precedente della macchina a stati a una nuova versione della macchina a stati. È possibile utilizzare questo script di esempio o aggiornarlo in base alle proprie esigenze.

Questo script mostra una distribuzione Canary per la distribuzione di una nuova versione di State Machine utilizzando un alias. I passaggi seguenti descrivono le attività eseguite dallo script:

1. Se il`publish_revision`il parametro è impostato su true, pubblica il più recenterevisioncome prossima versione della macchina a stati. Questa versione diventa la nuova versione live se la distribuzione ha esito positivo.

   Se si imposta il`publish_revision`parametro su false, lo script distribuisce l'ultima versione pubblicata della macchina a stati.
2. Crea un alias se non esiste ancora. Se l'alias non esiste, indirizza il 100% del traffico di questo alias alla nuova versione, quindi esci dallo script.
3. Aggiorna la configurazione di routing dell'alias per spostare una piccola percentuale di traffico dalla versione precedente alla nuova versione. Hai impostato questa percentuale canaria con`canary_percentage`parametro.
4. Per impostazione predefinita, monitora il configurabileCloudWatchallarmi ogni 60 secondi. Se uno di questi allarmi si attiva, ripristina immediatamente l'implementazione indirizzando il 100% del traffico alla versione precedente.

   Dopo ogni intervallo di tempo, in secondi, definito in`alarm_polling_interval`, continua a monitorare gli allarmi. Continuare il monitoraggio fino all'intervallo di tempo definito in`canary_interval_seconds`è passato.
5. Se non sono stati attivati allarmi durante`canary_interval_seconds`, sposta il 100% del traffico verso la nuova versione.
6. Se la nuova versione viene distribuita correttamente, elimina tutte le versioni precedenti al numero specificato nel`history_max`parametro.

```
#!/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!
```

Usa ilAWSSDK per implementare una nuova versione di state machine

Lo script di esempio all'indirizzoaws-stepfunctions-examplesmostra come usare ilAWSSDK per Python per spostare gradualmente il traffico da una versione precedente a una nuova versione di una macchina a stati. È possibile utilizzare questo script di esempio o aggiornarlo in base alle proprie esigenze.

Lo script mostra le seguenti strategie di implementazione:

- Canarie— Sposta il traffico in due incrementi.

Nel primo incremento, una piccola percentuale di traffico, ad esempio, il 10% viene spostata verso la nuova versione. Nel secondo incremento, prima che venga superato un intervallo di tempo specificato in secondi, il traffico rimanente viene spostato sulla nuova versione. Il passaggio alla nuova versione per il traffico residuo avviene solo in caso contrarioCloudWatchgli allarmi vengono attivati durante l'intervallo di tempo specificato.

- Lineare o rotolante— Sposta il traffico verso la nuova versione con incrementi uguali con un numero uguale di secondi tra ogni incremento.

  Ad esempio, se si specifica la percentuale di incremento come**20**con un`--interval`di**600**in pochi secondi, questa implementazione aumenta il traffico del 20% ogni 600 secondi fino a quando la nuova versione non riceve il 100% del traffico.

  Questa distribuzione ripristina immediatamente la nuova versione, se presenteCloudWatchgli allarmi sono attivati.
- Tutto in una volta o blu/verde— Sposta immediatamente il 100% del traffico verso la nuova versione. Questa distribuzione monitora la nuova versione e la ripristina automaticamente alla versione precedente, se presenteCloudWatchgli allarmi sono attivati.

UsoAWS CloudFormationper implementare una nuova versione della macchina a stati

Quanto segueCloudFormationesempio di modello pubblica due versioni di una macchina a stati denominata*MyStateMachine*. Crea un alias denominato*PROD*, che rimanda a entrambe queste versioni e quindi distribuisce la versione2.

In questo esempio, il 10 percento del traffico viene spostato verso la versione2ogni cinque minuti fino a quando questa versione riceve il 100% del traffico. Questo esempio mostra anche come è possibile impostareCloudWatchallarmi. Se uno qualsiasi degli allarmi che hai impostato, entra nelALARMstato, la distribuzione fallisce e viene ripristinata immediatamente.

```
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
```

Esecuzioni in Step Functions

L'esecuzione di una macchina a stati si verifica quando una macchina a stati AWS Step Functions viene eseguita e completa i rispettivi task. Ogni macchina a stati Step Functions può avere più esecuzioni simultanee, che è possibile avviare dalla console Step Functions o utilizzando gli AWS SDK, le azioni dell'API Step Functions o il (). AWS Command Line Interface AWS CLI Un'esecuzione riceve input JSON e genera output JSON. È possibile avviare un'esecuzione di Step Functions nei seguenti modi:

- Chiama l'operazione API StartExecution.
- Inizia una nuova esecuzione nella console Step Functions.
- Usa Amazon EventBridge per avviare un'esecuzione in risposta a un evento.
- Utilizzalo Amazon EventBridge Scheduler per avviare l'esecuzione di una macchina a stati in base a una pianificazione.
- Avvia un'esecuzione con Amazon API Gateway.
- Avvia l'esecuzione di un flusso di lavoro annidato da uno stato di Task.

Per ulteriori informazioni sui diversi modi di lavorare con Step Functions, consulta Development Options.

Avvio delle esecuzioni di flusso di lavoro da uno stato di task

AWS Step Functions può avviare le esecuzioni di flusso di lavoro direttamente da uno stato `Task` di una macchina a stati. In questo modo puoi suddividere i flussi di lavoro in macchine a stati più piccole e avviare le esecuzioni di queste altre macchine a stati. Avviando queste nuove esecuzioni di flusso di lavoro puoi:

- Separare il flusso di lavoro di livello superiore dai flussi di lavoro specifici del task di livello inferiore.
- Evitare gli elementi ripetitivi chiamando una macchina a stati separata più volte.
- Creare una libreria di flussi di lavoro riutilizzabili modulari per uno sviluppo più rapido.
- Ridurre la complessità e facilitare la modifica e la risoluzione dei problemi delle macchine a stati.

Step Functions può avviare queste esecuzioni del flusso di lavoro richiamando la propria API come servizio integrato. È sufficiente chiamare l'operazione API `StartExecution` dallo stato `Task` e passare i parametri necessari. Puoi chiamare l'API Step Functions utilizzando uno qualsiasi dei modelli di integrazione dei servizi.

> **Tip**
>
> Per implementare un esempio di flusso di lavoro annidato nel tuoAccount AWS, consulta Modulo 13 - Flussi di lavoro Nested Express.

Per avviare una nuova esecuzione di una macchina a stati, utilizzate uno `Task` stato simile all'esempio seguente:

```
{
   "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
}
```

Questo stato `Task` avvia una nuova esecuzione della macchina a stati `HelloWorld` e passa il commento JSON come input.

> **Note**
>
> Le quote dell'operazione API `StartExecution` possono condizionare il numero di esecuzioni che è possibile avviare. Utilizza `Retry` su `StepFunctions.ExecutionLimitExceeded` per assicurarti che l'esecuzione venga avviata. Consulta quanto segue.
>
> - Quote relative alla limitazione delle azioni delle API
> - Gestione degli errori in Step Functions

Associazione di esecuzioni di flusso di lavoro

Per associare un'esecuzione di flusso di lavoro avviata all'esecuzione che l'ha avviata, passa l'ID di esecuzione dall'oggetto contesto all'input di esecuzione. Puoi accedere all'ID dall'oggetto contesto del tuo stato `Task` in un'esecuzione in corso. Passa l'ID di esecuzione aggiungendo `.$` al nome del parametro e facendo riferimento all'ID nell'oggetto contesto con `$$.Execution.Id`.

```
"AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID.$": "$$.Execution.Id"
```

Puoi utilizzare un parametro speciale denominato `AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID` all'avvio di un'esecuzione. Se inclusa, questa associazione fornisce collegamenti nella sezione Dettagli della fase della console Step Functions. Se vengono forniti, puoi facilmente tracciare le esecuzioni dei tuoi flussi di lavoro dall'avvio all'esecuzione di flusso di lavoro avviata. Utilizzando l'esempio precedente, associa l'ID di esecuzione all'esecuzione avviata della macchina a stati `HelloWorld` come segue.

```
{
   "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
}
```

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Uso di altri servizi
- Passa i parametri a un'API di servizio
- Accesso all'oggetto contestuale
- AWS Step Functions

Utilizzo di Amazon EventBridge Scheduler con AWS Step Functions

Amazon EventBridge Scheduler è uno strumento di pianificazione senza server che consente di creare, eseguire e gestire attività da un unico servizio gestito centralizzato. Con EventBridge Scheduler, puoi creare pianificazioni utilizzando le espressioni cron e rate per schemi ricorrenti o configurare chiamate una tantum. Puoi configurare finestre temporali flessibili per la consegna, definire limiti per nuovi tentativi e impostare il tempo massimo di conservazione per invocazioni API non riuscite.

Ad esempio, con EventBridge Scheduler, è possibile avviare l'esecuzione di una macchina a stati in base a una pianificazione quando si verifica un evento relativo alla sicurezza o per automatizzare un processo di elaborazione dati.

Questa pagina spiega come utilizzare EventBridge Scheduler per avviare l'esecuzione di una macchina a stati Step Functions in base a una pianificazione.

Argomenti

- Configurare il ruolo di esecuzione
- Creare una pianificazione.
- Risorse correlate

Configurare il ruolo di esecuzione

Quando si crea una nuova EventBridge pianificazione, Scheduler deve disporre dell'autorizzazione per richiamare l'operazione API di destinazione per conto dell'utente. Concedi queste autorizzazioni a EventBridge Scheduler utilizzando un ruolo di esecuzione. La policy di autorizzazione collegata al ruolo di esecuzione della pianificazione definisce le autorizzazioni necessarie. Queste autorizzazioni dipendono dall'API di destinazione che si desidera che EventBridge Scheduler richiami.

Quando si utilizza la console EventBridge Scheduler per creare una pianificazione, come nella procedura seguente, EventBridge Scheduler imposta automaticamente un ruolo di esecuzione in base all'obiettivo selezionato. Se si desidera creare una pianificazione utilizzando uno degli EventBridge Scheduler SDK AWS CLIAWS CloudFormation, oppure è necessario disporre di un ruolo di esecuzione esistente che conceda le autorizzazioni richieste da EventBridge Scheduler per richiamare una destinazione. Per ulteriori informazioni sull'impostazione manuale di un ruolo di esecuzione per la pianificazione, consulta Configurazione di un ruolo di esecuzione nella Guida per l'utente di Scheduler. EventBridge

Creare una pianificazione.

Per creare una pianificazione utilizzando la console

1. Apri la console Amazon EventBridge Scheduler all'indirizzo https://console.aws.amazon.com/scheduler/home.
2. Nella pagina Pianificazioni, scegli Crea pianificazione.

3. Nella pagina Specifica i dettagli della pianificazione, nella sezione Nome e descrizione della pianificazione, effettua le seguenti operazioni:

   a. Per Nome pianificazione, inserisci un nome per la pianificazione. Ad esempio, **`MyTestSchedule`**.

   b. (Facoltativo) Per Descrizione, inserisci una descrizione per la pianificazione. Ad esempio, **`My first schedule`**.

   c. Per Gruppo di pianificazioni, scegli un gruppo di pianificazioni dall'elenco a discesa. Se non hai un gruppo, scegli predefinito. Per creare un gruppo di pianificazioni, scegli crea la tua pianificazione.

      I gruppi di pianificazione vengono utilizzati per aggiungere tag a gruppi di pianificazioni.

4. • Scegli le opzioni di pianificazione.

Ricorrenza	Esegui questa operazione…	
Pianificazione una tantum Una pianificazione unica richiama una destinazione solo una volta alla data e all'ora specificate.	Per Data e ora, effettua le seguenti operazioni: • Inserisci una data valida in formato `YYYY/MM/DD`. • Inserisci un timestamp in formato `hh:mm` 24 ore. • Per Fuso orario, scegli il fuso orario.	
Pianificazione ricorrente Una pianificazione ricorrente richiama una destinazione con una frequenza specificata utilizzando un'espressione cron o un'espressione rate.	a. Per Tipo di pianificazione, esegui una delle seguenti operazioni: • Per utilizzare un'espressione cron per definire la pianificazione, scegli Pianificazione basata su cron e	

Ricorrenza	Esegui questa operazione…	
	inserisci l'espressione cron. • Per utilizzare un'espressione di frequenza per definire la pianificazione, scegli Pianificazione basata su frequenza e inserisci l'espressione di frequenza. Per ulteriori informazioni sulle espressioni cron e rate, consulta Schedule types on EventBridge Scheduler nella Amazon EventBridge Scheduler User Guide. b. Per Finestra temporale flessibile, scegli Disattivata per disattivare l'opzione o scegli una delle finestre temporali predefinite. Ad esempio, se scegli 15 minuti e imposti una pianificazione ricorrente per il richiamo della destinazione ogni ora, la pianificazione viene eseguita entro 15 minuti dall'inizio di ogni ora.	

5. (Facoltativo) Se hai scelto Pianificazione ricorrente nel passaggio precedente, nella sezione Intervallo di tempo effettua le seguenti operazioni:

   a. Per Fuso orario, scegli un fuso orario.

   b. Per Data e ora di inizio, inserisci una data valida in formato `YYYY/MM/DD`, quindi specifica un timestamp in formato `hh:mm` 24 ore.

   c. Per Data e ora di fine, inserisci una data valida in formato `YYYY/MM/DD`, quindi specifica un timestamp in formato `hh:mm` 24 ore.

6. Seleziona Successivo.

7. Nella pagina Seleziona destinazione, scegli l'operazione AWS API richiamata da Scheduler: EventBridge

   a. Scegli. AWS Step Functions StartExecution

   b. Nella StartExecutionsezione, seleziona una macchina a stati o scegli Crea nuova macchina a stati.

      Al momento, non è possibile eseguire i flussi di lavoro Synchronous Express in base a una pianificazione.

   c. Inserisci un payload JSON per l'esecuzione. Anche se la tua macchina a stati non richiede alcun payload JSON, devi comunque includere l'input in formato JSON, come mostrato nell'esempio seguente.

      ```
      {
          "Comment": "sampleJSONData"
      }
      ```

8. Seleziona Successivo.

9. Nella pagina Settings (Impostazioni), eseguire le operazioni descritte di seguito.

   a. Per attivare la pianificazione, in Stato della pianificazione, attiva Abilita pianificazione.

   b. Per configurare una policy per nuovi tentativi per una pianificazione, in Policy per nuovi tentativi e DLQ (dead-letter queue) effettua le seguenti operazioni:

      - Attiva/disattiva Riprova.
      - Per Età massima dell'evento, inserite il numero massimo di ore e minuti in cui EventBridge Scheduler deve conservare un evento non elaborato.
      - La durata massima è 24 ore.

- Per Numero massimo di tentativi, inserisci il numero massimo di volte in cui EventBridge Scheduler riprova la pianificazione se la destinazione restituisce un errore.

  Il valore massimo è 185 tentativi.

Con le politiche di ripetizione dei tentativi, se una pianificazione non riesce a richiamare l'obiettivo, EventBridge Scheduler esegue nuovamente la pianificazione. Se configurato, è necessario impostare il tempo di conservazione massimo e i nuovi tentativi per la pianificazione.

c. Scegli dove EventBridge Scheduler archivia gli eventi non consegnati.

Opzione Dead-letter queue (DLQ)	Esegui questa operazione…
Non conservare	Scegliere None (Nessuno).
Memorizza l'evento nello stesso Account AWS in cui crei la pianificazione	a. Scegli Seleziona una coda Amazon SQS nel mio Account AWS come DLQ. b. Scegli il nome della risorsa Amazon (ARN) della coda di Amazon SQS.
Memorizza l'evento in un Account AWS diverso da quello in cui crei la pianificazione	a. Scegli Specifica una coda Amazon SQS in un altro Account AWS come DLQ. b. Inserisci il nome della risorsa Amazon (ARN) della coda di Amazon SQS.

d. Per utilizzare una chiave gestita dal cliente per crittografare l'input di destinazione, in Crittografia scegli Personalizza le impostazioni di crittografia (avanzate).

Se scegli questa opzione, inserisci l'ARN di una chiave KMS esistente scegli Crea una AWS KMS key per accedere alla console AWS KMS. Per ulteriori informazioni su come EventBridge Scheduler crittografa i dati inattivi, consulta Encryption at rest nella Amazon EventBridge Scheduler User Guide.

e. Per fare in modo che EventBridge Scheduler crei un nuovo ruolo di esecuzione per te, scegli Crea nuovo ruolo per questa pianificazione. Inserisci, quindi, un nome per Nome ruolo. Se scegli questa opzione, EventBridge Scheduler assegna al ruolo le autorizzazioni necessarie per la destinazione basata sul modello.

10. Seleziona Successivo.
11. Nella pagina Rivedi e crea pianificazione, rivedi i dettagli della pianificazione. In ogni sezione, scegli Modifica per tornare a tale passaggio e modificarne i dettagli.
12. Scegli Crea pianificazione.

    Puoi visualizzare un elenco delle pianificazioni nuove ed esistenti nella pagina Pianificazioni. Nella colonna Stato, accertati che la nuova pianificazione sia Abilitata.

Per confermare che EventBridge Scheduler abbia richiamato la macchina a stati, controlla i log Amazon della macchina a stati. CloudWatch

Risorse correlate

Per ulteriori informazioni su EventBridge Scheduler, consulta quanto segue:

- EventBridge Guida per l'utente di Scheduler
- EventBridge Riferimento all'API Scheduler
- EventBridge Prezzi Scheduler

Esecuzioni Standard ed Express Workflow nella console

Quando crei una macchina a stati, selezioni un tipo Standard o Express. Il tipo predefinito per le macchine a stati è Standard. Una macchina a stati il cui tipo è Standard viene denominata flusso di lavoro Standard e una macchina a stati il cui tipo è Express viene chiamata flusso di lavoro Express.

Sia per i flussi di lavoro Standard che Express, si definisce la macchina a stati utilizzando. Amazon States Language Le esecuzioni delle macchine a stati si comporteranno in modo diverso a seconda del tipo selezionato.

> **Important**
>
> Il tipo scelto non può essere modificato dopo aver creato la macchina a stati.

Per ulteriori informazioni sui flussi di lavoro Standard ed Express, consultaFlussi di lavoro Standard e Express.

La cronologia delle esecuzioni dei flussi di lavoro Standard viene registrata in Step Functions, mentre la cronologia delle esecuzioni dei flussi di lavoro Express non viene registrata in Step Functions. Per registrare la cronologia dell'esecuzione di un flusso di lavoro Express, devi configurarlo per inviare i log ad Amazon CloudWatch. Per ulteriori informazioni, consulta Registrazione utilizzandoCloudWatchRegistri.

Una volta configurata la registrazione su un flusso di lavoro Express, è possibile visualizzarne le esecuzioni nella console Step Functions. L'esperienza della console per visualizzare le esecuzioni dei flussi di lavoro Express e le esecuzioni dei flussi di lavoro Standard è simile, ad eccezione delle seguenti differenze e limitazioni.

> **Note**
>
> Poiché i dati di esecuzione per i flussi di lavoro Express vengono visualizzati utilizzando CloudWatch Logs Insights, la scansione dei registri comporterà dei costi. Per impostazione predefinita, il gruppo di log elenca solo le esecuzioni completate nelle ultime tre ore. Se si specifica un intervallo di tempo più ampio che include più eventi di esecuzione, i costi aumenteranno. Per ulteriori informazioni, consulta Vending Logs nella scheda Logs della pagina CloudWatch Prezzi e. Registrazione utilizzandoCloudWatchRegistri

Indice

- Differenze nell'esperienza della console
- Considerazioni e limitazioni per la visualizzazione delle esecuzioni del flusso di lavoro Express

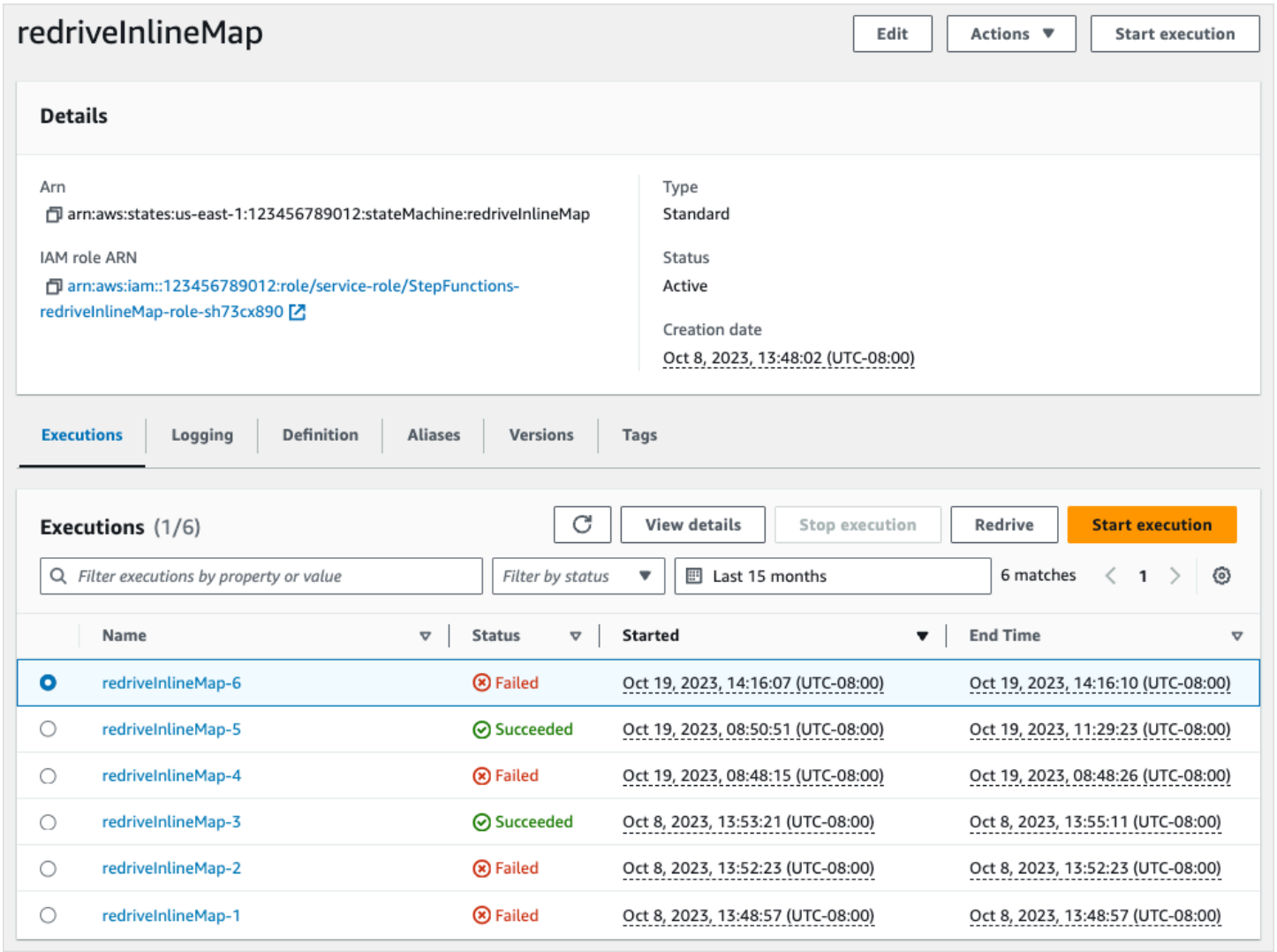
Differenze nell'esperienza della console

Per tutti i flussi di lavoro Standard ed Express, puoi visualizzare i dettagli, come la macchina a stati e il relativo ruolo IAM ARN, nella pagina dei dettagli della macchina a stati della console Step Functions.

Nella pagina dei dettagli della macchina a stati, puoi anche visualizzare un elenco delle cronologie di esecuzione della tua macchina a stati nella scheda Esecuzioni. Utilizza la casella Filtra le esecuzioni per proprietà o valore per cercare un'esecuzione, una versione o un alias specifici della macchina a stati scelta. Utilizza il menu a discesa Tutto per filtrare le cronologie di esecuzione in base al loro stato. Puoi anche scegliere una cronologia di esecuzione e selezionare il pulsante Visualizza dettagli per aprire la relativa pagina dei dettagli di esecuzione.

Flussi di lavoro standard

Le cronologie di esecuzione per i flussi di lavoro standard sono sempre disponibili per le esecuzioni completate negli ultimi 90 giorni.



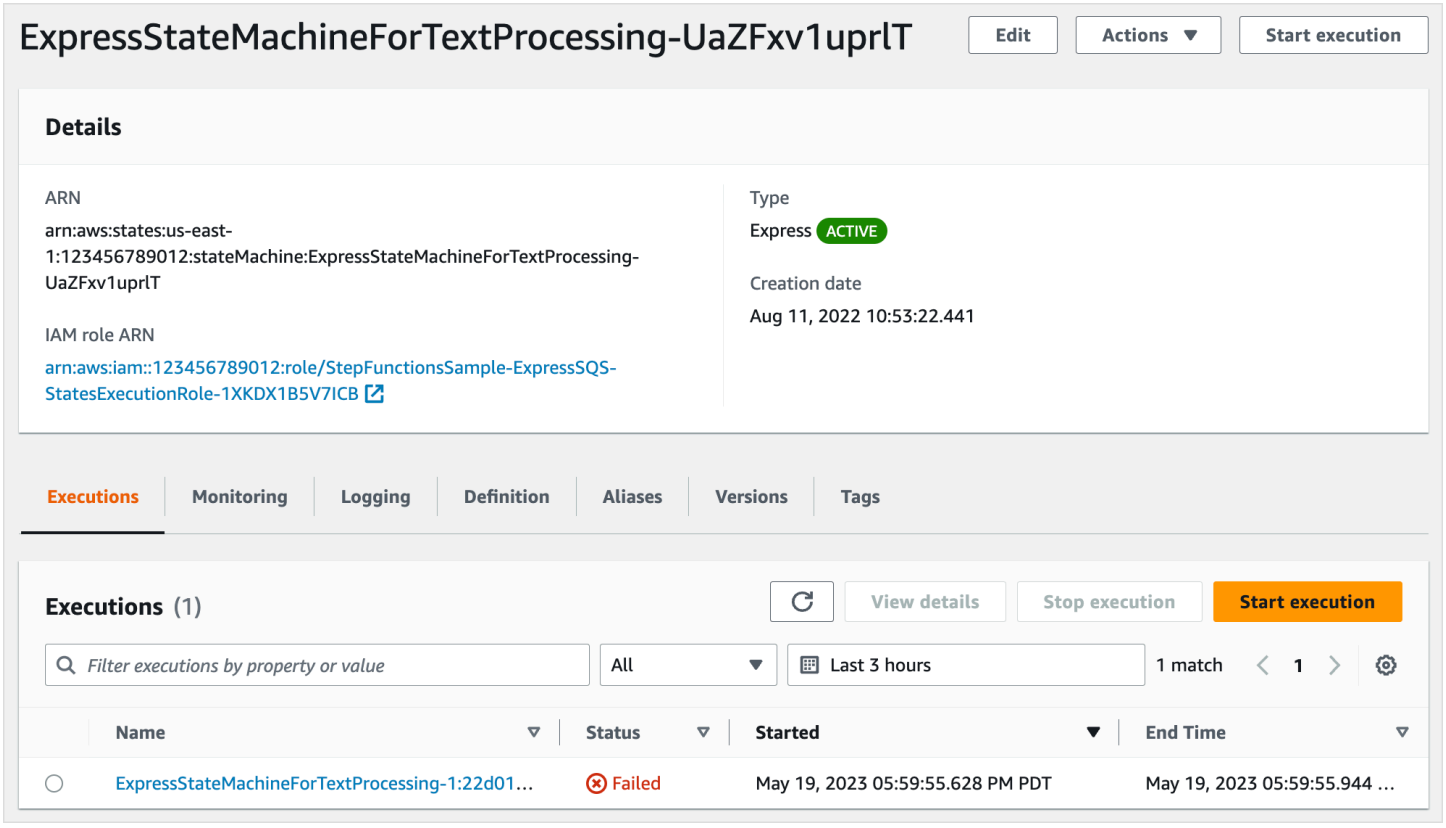
Flussi di lavoro Express

Per visualizzare la cronologia di esecuzione per i flussi di lavoro Express, la console Step Functions recupera i dati di registro raccolti tramite un gruppo di log CloudWatch Logs.

È inoltre necessario abilitare la nuova esperienza della console per visualizzare le esecuzioni dei flussi di lavoro Express. Per fare ciò, scegli il pulsante Abilita visualizzato all'interno del banner nella scheda Esecuzioni. Una volta scelto questo pulsante, non verrà più visualizzato.

> **Tip**
>
> Per passare dall'attivazione alla disabilitazione dell'esperienza della console, utilizza il pulsante di attivazione della cronologia delle esecuzioni rapide.

Le cronologie delle esecuzioni completate nelle ultime tre ore sono disponibili per impostazione predefinita. È possibile modificare questo intervallo di tempo o specificare un intervallo personalizzato. Se si specifica un intervallo di tempo più ampio che include più eventi di esecuzione, il costo di scansione dei log aumenterà. Per ulteriori informazioni, consulta Vending Logs nella scheda Logs della pagina Prezzi e. CloudWatch Registrazione utilizzandoCloudWatchRegistri



Considerazioni e limitazioni per la visualizzazione delle esecuzioni del flusso di lavoro Express

Quando visualizzi le esecuzioni dei flussi di lavoro Express sulla console Step Functions, tieni presenti le considerazioni e le limitazioni seguenti.

- La disponibilità dei dettagli di esecuzione del flusso di lavoro Express si basa su Amazon CloudWatch Logs
- I dettagli sull'esecuzione del flusso di lavoro Partial Express sono disponibili se il livello di registrazione è ERROR o FATAL
- La definizione della macchina a stati di un'esecuzione precedente non può essere visualizzata una volta aggiornata

La disponibilità dei dettagli di esecuzione del flusso di lavoro Express si basa su Amazon CloudWatch Logs

> **Note**
>
> Se non abiliti la nuova esperienza della console per visualizzare le esecuzioni dei flussi di lavoro Express, le cronologie di esecuzione e i dettagli di esecuzione corrispondenti non sono disponibili nella console Step Functions. Per abilitare la nuova esperienza di console, scegli il pulsante Abilita visualizzato all'interno del banner nella scheda Esecuzioni.

Per i flussi di lavoro Express, la cronologia di esecuzione e le informazioni dettagliate sull'esecuzione vengono raccolte tramite CloudWatch Logs Insights. Queste informazioni vengono conservate nel gruppo di log CloudWatch Logs specificato al momento della creazione della macchina a stati. La cronologia delle esecuzioni della macchina a stati viene mostrata nella scheda Executions della console Step Functions. Informazioni dettagliate su ogni esecuzione della macchina a stati vengono visualizzate nella pagina dei dettagli di esecuzione relativa all'esecuzione scelta.

> **Warning**
>
> Se elimini il flusso di lavoro CloudWatch Logs for an Express, non verrà elencato nella scheda Esecuzioni.

Si consiglia di utilizzare il livello di registro predefinito ALL per registrare tutti i tipi di eventi di esecuzione. È possibile aggiornare il livello di registro come richiesto per le macchine a stati esistenti quando le si modifica. Per ulteriori informazioni, consultare Registrazione utilizzandoCloudWatchRegistri e Livelli di log.

I dettagli sull'esecuzione del flusso di lavoro Partial Express sono disponibili se il livello di registrazione è ERROR o FATAL

Per impostazione predefinita, il livello di registrazione per le esecuzioni del flusso di lavoro Express è impostato su ALL. Se si modifica il livello di registro, le cronologie di esecuzione e i dettagli di esecuzione delle esecuzioni completate non subiranno alcuna modifica. Tuttavia, tutte le nuove esecuzioni emetteranno registri in base al livello di registro aggiornato. Per ulteriori informazioni, consultare Registrazione utilizzandoCloudWatchRegistri e Livelli di log.

Ad esempio, se si modifica il livello di registro da ALL a ERROR o FATAL, la scheda Executions della console Step Functions elenca solo le esecuzioni non riuscite. Nella scheda Visualizzazione eventi, la console mostra solo i dettagli degli eventi relativi ai passaggi della macchina a stati che non sono riusciti.

Si consiglia di utilizzare il livello di registro predefinito ALL per registrare tutti i tipi di eventi di esecuzione. È possibile aggiornare il livello di registro come richiesto per le macchine a stati esistenti quando si modifica la macchina a stati.

La definizione della macchina a stati di un'esecuzione precedente non può essere visualizzata una volta aggiornata

Le definizioni delle macchine a stati per le esecuzioni precedenti non vengono archiviate per i flussi di lavoro Express. Se si modifica la definizione della macchina a stati, è possibile visualizzare solo la definizione della macchina a stati per le esecuzioni utilizzando la definizione più recente.

Ad esempio, se rimuovete uno o più passaggi dalla definizione della macchina a stati, Step Functions rileva una mancata corrispondenza tra la definizione e gli eventi di esecuzione precedenti. Poiché le definizioni precedenti non sono memorizzate per i flussi di lavoro Express, Step Functions non può visualizzare la definizione della macchina a stati per le esecuzioni eseguite su una versione precedente della definizione della macchina a stati. Di conseguenza, le schede Execution input & output, Definition, Graph view e Table view non sono disponibili per le esecuzioni eseguite su versioni precedenti della definizione di una macchina a stati.

Visualizzazione e debug delle esecuzioni sulla console Step Functions

La pagina Execution Details della console Step Functions presenta informazioni sulle esecuzioni passate e in corso di macchine a stati per i flussi di lavoro Standard ed Express. Queste informazioni vengono visualizzate in un formato dashboard. Ad esempio, puoi trovare la definizione di Amazon States Language della macchina a stati, il suo stato di esecuzione, l'ARN e il numero totale di transizioni di stato. Puoi anche visualizzare i dettagli di esecuzione per ogni singolo stato nella macchina a stati.

Indice

- Pagina dei dettagli di esecuzione: panoramica dell'interfaccia
  - Riepilogo dell'esecuzione
  - Messaggio di errore
  - Modalità di visualizzazione
  - Dettagli del passaggio
  - Eventi
- Tutorial: Esame delle esecuzioni delle macchine a stati utilizzando la console Step Functions
  - Fase 1: Creare e testare le funzioni Lambda richieste
  - Fase 2: Creare ed eseguire la macchina a stati
  - Fase 3: Visualizzare i dettagli di esecuzione della macchina a stati
  - Fase 4: Esplora le diverse modalità di visualizzazione

Pagina dei dettagli di esecuzione: panoramica dell'interfaccia

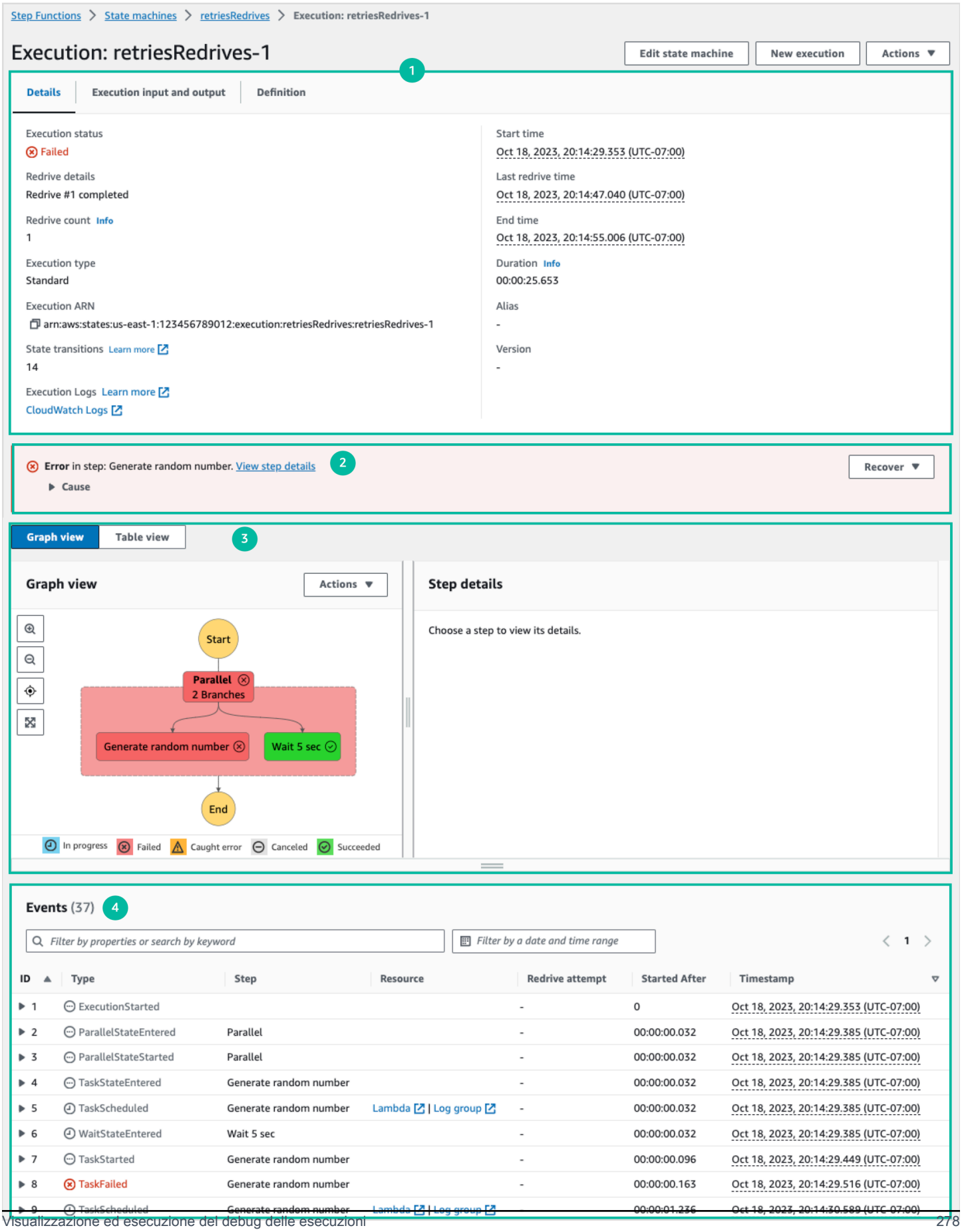
Puoi trovare i dettagli di tutte le esecuzioni in corso e passate di macchine a stati per i flussi di lavoro Standard ed Express nella pagina Dettagli di esecuzione. Se hai specificato un ID di esecuzione all'inizio dell'esecuzione, questa pagina è intitolata con quell'ID di esecuzione. Altrimenti, è intitolato con l'ID di esecuzione univoco che Step Functions genera automaticamente per te.

Oltre alle metriche di esecuzione, la pagina Dettagli di esecuzione fornisce le seguenti opzioni per la gestione della macchina a stati e della relativa esecuzione:

Button	Scegliete questo pulsante per:
Modifica macchina a stati	Modifica la definizione di Amazon States Language della tua macchina a stati.
Nuova esecuzione	Inizia una nuova esecuzione della tua macchina a stati.
Operazioni	Fornisce le seguenti opzioni tra cui scegliere: • Interrompi esecuzione: interrompe un'esecuzione in corso. Questa opzione non è disponibile per le esecuzioni completate. • Redrive— Redrive esecuzioni di flussi di lavoro standard che non sono stati completati correttamente negli ultimi 14 giorni. Queste includono esecuzioni non riuscite, interrotte o scadute. Per ulteriori informazioni, consulta Redrivingesecuzioni. • Esporta: esporta i dettagli di esecuzione in formato JSON per condividerli con altri o eseguire analisi offline. • Invia feedback: condividi feedback sull'interfaccia.

> **Visualizzazione delle esecuzioni iniziate con una versione o un alias**
>
> Puoi anche visualizzare le esecuzioni iniziate con una versione o un alias nella console Step Functions. Per ulteriori informazioni, consulta Elencare le esecuzioni per versioni e alias.

La pagina della console Execution Details contiene le seguenti sezioni:


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. Riepilogo dell'esecuzione
2. Messaggio di errore
3. Modalità di visualizzazione
4. Dettagli del passaggio
5. Eventi

Riepilogo dell'esecuzione

La sezione di riepilogo dell'esecuzione viene visualizzata nella parte superiore della pagina Dettagli di esecuzione. Questa sezione fornisce una panoramica dei dettagli di esecuzione del flusso di lavoro. Queste informazioni sono suddivise tra le tre schede seguenti:

Dettagli

Mostra informazioni, come lo stato dell'esecuzione, l'ARN e i timestamp per l'ora di inizio e fine dell'esecuzione. È inoltre possibile visualizzare il conteggio totale delle transizioni di stato avvenute durante l'esecuzione della macchina a stati. Puoi anche visualizzare i link per la mappa di tracciamento X-Ray e Amazon CloudWatch Execution Logs se hai abilitato il tracciamento o i log per la tua macchina a stati.

Se l'esecuzione della tua macchina a stati è stata avviata da un'altra macchina a stati, puoi visualizzare il link per la macchina a stati principale in questa scheda.

Se l'esecuzione della macchina a stati è avvenuta redriven, questa scheda mostra le informazioni redrive correlate, ad esempio il Redriveconteggio.

Input e output dell'esecuzione

Mostra l'input e l'output di esecuzione della macchina a stati side-by-side.

Definizione

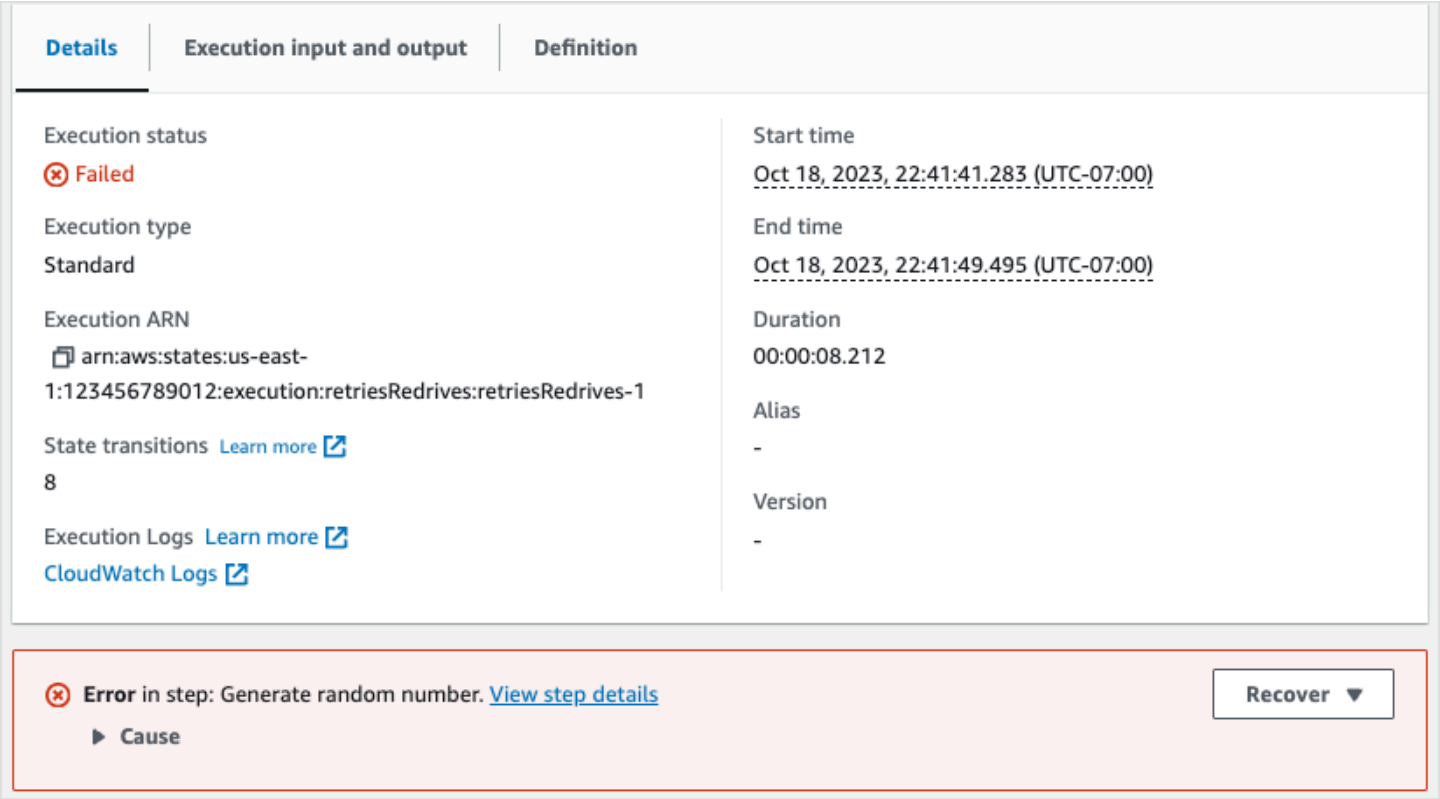
Mostra la definizione di Amazon States Language della macchina a stati.

Messaggio di errore

Se l'esecuzione della macchina a stati non è riuscita, nella pagina Dettagli di esecuzione viene visualizzato un messaggio di errore. Scegli Causa o Visualizza i dettagli del passaggio nel messaggio di errore per visualizzare il motivo dell'errore di esecuzione o il passaggio che ha causato l'errore.

Se scegliete Visualizza dettagli passo, Step Functions evidenzia il passaggio che ha causato l'errore nelle schede Dettagli passo, Vista grafico e Vista tabella. Se il passaggio è uno stato Task, Map o Parallel per il quale sono stati definiti nuovi tentativi, nel riquadro dei dettagli del passaggio viene visualizzata la scheda Riprova relativa al passaggio. Inoltre, se è in corso redriven l'esecuzione, è possibile visualizzare i nuovi tentativi e i dettagli di redrive esecuzione nella redrives scheda Ritentativi e del riquadro dei dettagli del passaggio.

Dal pulsante a discesa Ripristina di questo messaggio di errore, puoi scegliere se eseguire redrive le esecuzioni non riuscite o iniziare una nuova esecuzione. Per ulteriori informazioni, consulta Redrivingesecuzioni.



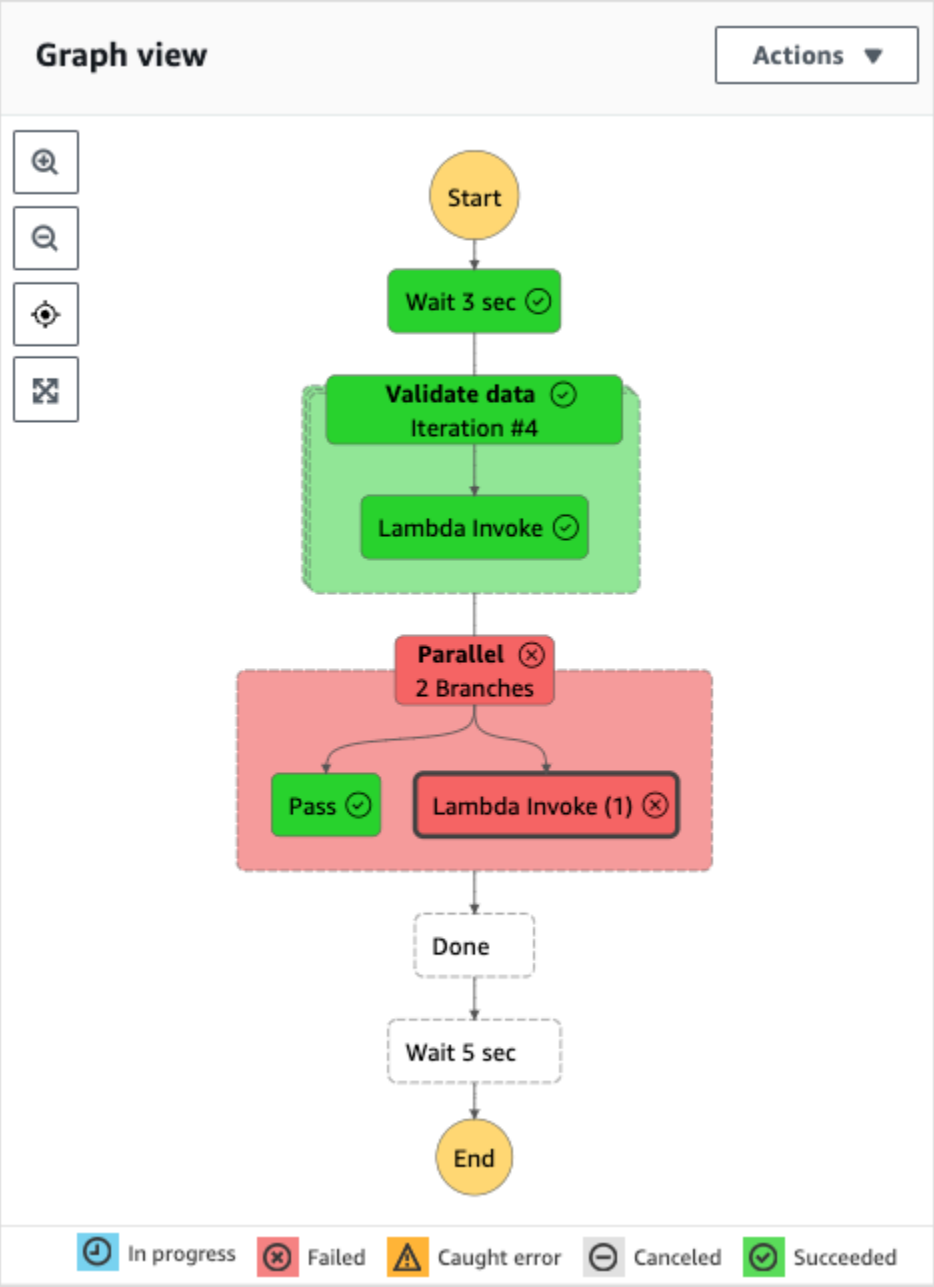
Modalità di visualizzazione

La sezione della modalità di visualizzazione contiene due diverse visualizzazioni per la macchina a stati. È possibile scegliere di visualizzare una rappresentazione grafica del flusso di lavoro, una tabella che delinea gli stati del flusso di lavoro o un elenco degli eventi associati all'esecuzione della macchina a stati:

> **Note**
>
> Scegliete una scheda per visualizzarne il contenuto.

Graph view

La modalità di visualizzazione Graph mostra una rappresentazione grafica del flusso di lavoro. Nella parte inferiore è inclusa una legenda che indica lo stato di esecuzione della macchina a stati. Contiene inoltre pulsanti che consentono di ingrandire, rimpicciolire, allineare al centro l'intero flusso di lavoro o visualizzare il flusso di lavoro in modalità a schermo intero.



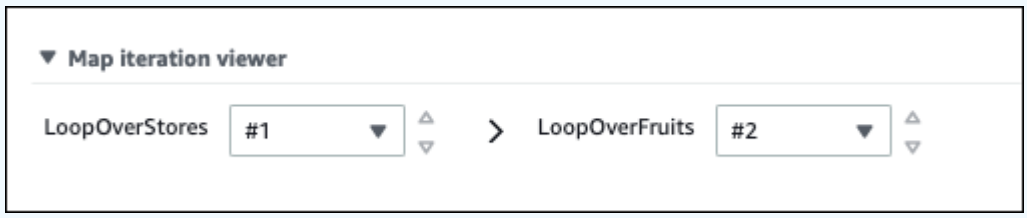
Da questa vista, è possibile scegliere qualsiasi fase del flusso di lavoro per visualizzare i dettagli sulla sua esecuzione nel componente Dettagli del passaggio. Quando scegliete un passaggio

nella vista Grafico, anche la vista Tabella mostra quel passaggio. Questo vale anche al contrario. Se scegliete un passaggio dalla vista Tabella, la vista Grafico mostra lo stesso passaggio.

Se la macchina a stati contiene uno `Map` stato, `Parallel` uno stato o entrambi, puoi visualizzarne i nomi nel flusso di lavoro nella vista Grafico. Inoltre, per quanto riguarda `Map` lo stato, la vista Graph consente di spostarsi tra diverse iterazioni dei dati di esecuzione dello stato della mappa. Ad esempio, se lo stato della mappa ha cinque iterazioni e desideri visualizzare i dati di esecuzione per la terza e la quarta iterazione, procedi come segue:

1. Scegli lo stato della mappa di cui desideri visualizzare i dati di iterazione.
2. Dal Map Iteration Viewer, scegli #2 dall'elenco a discesa per la terza iterazione. Questo perché le iterazioni vengono contate da zero. Allo stesso modo, scegli #3 dall'elenco a discesa per la quarta iterazione dello stato della mappa.

   In alternativa, usa i
   ▽
   controlli
   △
   and per spostarti tra le diverse iterazioni dello stato della mappa.

   > **Note**
   >
   > Se la macchina a stati contiene `Map` stati annidati, gli elenchi a discesa per le iterazioni `Map` dello stato principale e figlio verranno visualizzati come mostrato nell'esempio seguente:
   >
   > 
   >

3. (Facoltativo) Se una o più iterazioni dello stato della mappa non sono state eseguite o l'esecuzione è stata interrotta, puoi visualizzarne i dati scegliendo i numeri di iterazione in Non riuscito o Interrotto nell'elenco a discesa.

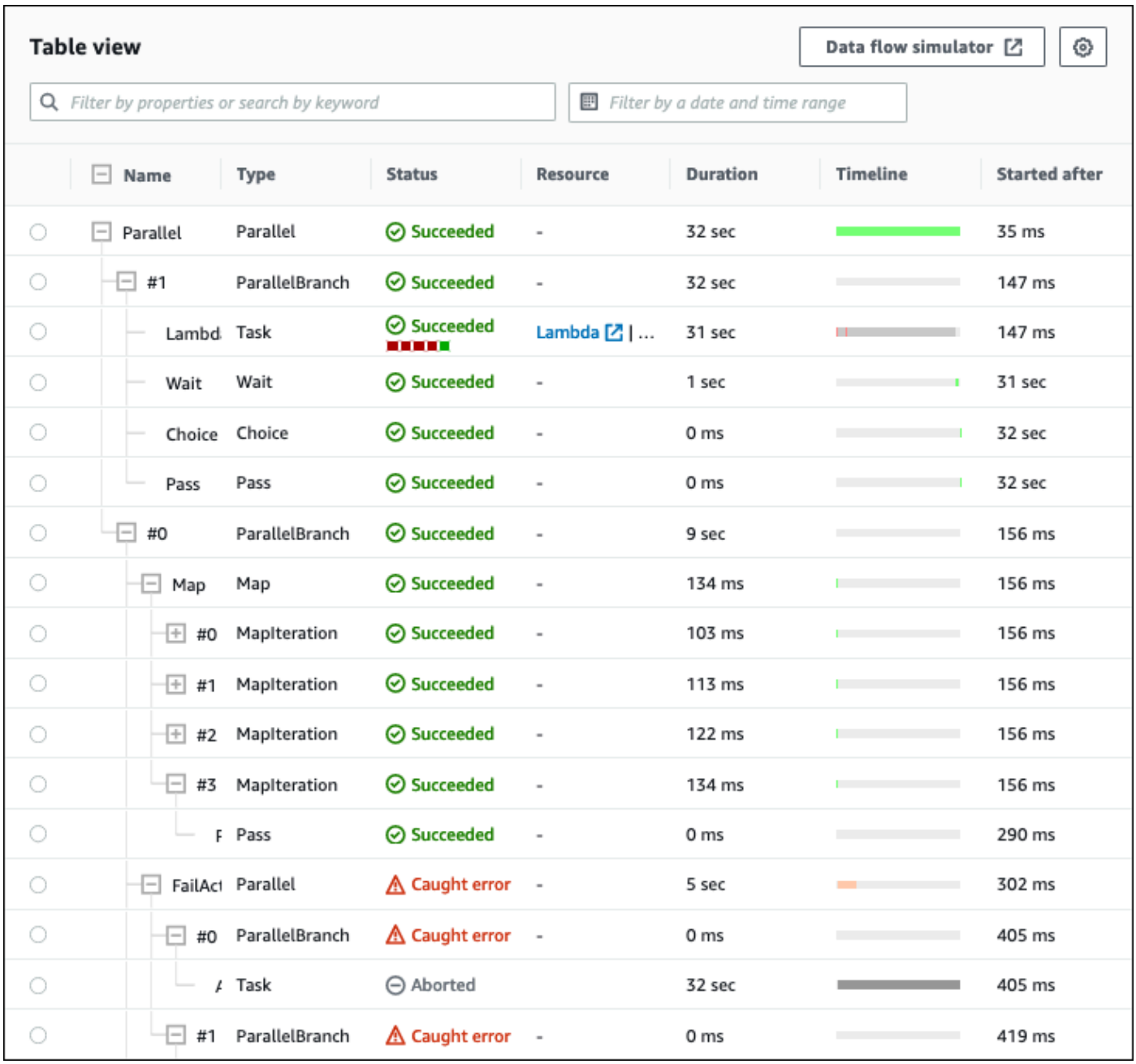
Infine, puoi utilizzare i pulsanti Esporta e Layout per esportare il grafico del flusso di lavoro come immagine SVG o PNG. Potete anche passare dalla visualizzazione orizzontale a quella verticale del flusso di lavoro.

Table view

La modalità di visualizzazione Tabella mostra una rappresentazione tabulare degli stati del flusso di lavoro. In questa modalità di visualizzazione, è possibile visualizzare i dettagli di ogni stato eseguito nel flusso di lavoro, incluso il nome, il nome di qualsiasi risorsa utilizzata (ad esempio una AWS Lambda funzione) e se lo stato è stato eseguito correttamente.

Da questa vista, è possibile scegliere qualsiasi stato del flusso di lavoro per visualizzare i dettagli sulla sua esecuzione nel componente Step details. Quando si sceglie un passaggio nella vista Tabella, anche la vista Grafico mostra tale passaggio. Questo vale anche al contrario. Se scegliete un passaggio dalla vista Grafico, la vista Tabella mostra lo stesso passaggio.

È inoltre possibile limitare la quantità di dati visualizzati nella modalità di visualizzazione Tabella applicando filtri alla vista. È possibile creare un filtro per una proprietà specifica, ad esempio Status o Redrivetry. Per ulteriori informazioni, consulta Tutorial: Esame delle esecuzioni delle macchine a stati utilizzando la console Step Functions.

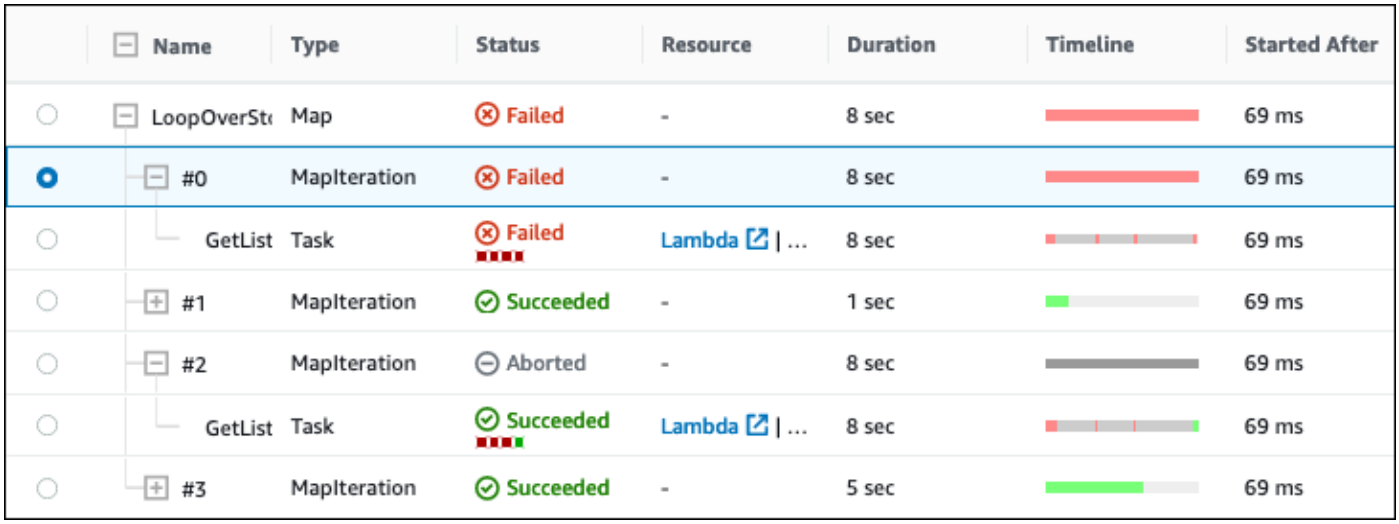


Per impostazione predefinita, questa modalità visualizza le colonne Nome, Tipo, Stato, Risorsa e Started After. È possibile configurare le colonne che si desidera visualizzare utilizzando la finestra di dialogo Preferenze. Le selezioni effettuate in questa finestra di dialogo persistono per le future esecuzioni di macchine a stati finché non vengono nuovamente modificate.

Se aggiungete la colonna Cronologia, viene mostrata la durata dell'esecuzione di ogni stato rispetto al runtime per l'intera esecuzione. Viene visualizzata come una sequenza temporale lineare codificata a colori. Questo può aiutarti a identificare eventuali problemi relativi alle prestazioni con l'esecuzione di uno stato specifico. I segmenti codificati a colori per ogni stato

sulla timeline consentono di identificare lo stato di esecuzione dello stato, ad esempio in corso, non riuscito o interrotto.

Ad esempio, se avete definito nuovi tentativi di esecuzione per uno stato nella vostra macchina a stati, questi tentativi vengono mostrati nella timeline. I segmenti rossi rappresentano i `Retry` tentativi falliti, mentre i segmenti di colore grigio chiaro rappresentano gli intervalli tra un tentativo e l'altro. `BackoffRate` `Retry`



Se la macchina a stati contiene uno stato, uno **`Map`** stato o entrambi, **`Parallel`** puoi visualizzarne i nomi nel flusso di lavoro in visualizzazione Tabella. Per `Parallel` gli stati `Map` e, la modalità di visualizzazione Tabella mostra i dati di esecuzione per le loro iterazioni e i rami paralleli come nodi all'interno di una vista ad albero. Puoi scegliere ogni nodo in questi stati per visualizzarne i dettagli individuali nella sezione Dettagli del passaggio. Ad esempio, puoi esaminare i dati per una specifica iterazione dello stato della mappa che ha causato il fallimento dello stato. Espandi il nodo per lo stato della mappa, quindi visualizza lo stato di ogni iterazione nella colonna Stato.

Dettagli del passaggio

La sezione dei dettagli del passaggio si apre sulla destra quando si sceglie uno stato nella vista Grafico o nella vista Tabella. Questa sezione contiene le seguenti schede, che forniscono informazioni approfondite sullo stato selezionato:

Input

Mostra i dettagli di input dello stato selezionato. Se c'è un errore nell'immissione, viene indicato con un

nell'intestazione della scheda. Inoltre, puoi visualizzare il motivo dell'errore in questa scheda.

Puoi anche scegliere il pulsante Visualizzazione avanzata per visualizzare il percorso di trasferimento dei dati di input man mano che i dati passano attraverso lo stato selezionato. Ciò consente di identificare in che modo l'input è stato elaborato quando uno o più campi, ad esempio `InputPathParameters`,`ResultSelector`, e `OutputPathResultPath`, sono stati applicati ai dati.

Output

Mostra l'output dello stato selezionato. Se c'è un errore nell'output, viene indicato con un

nell'intestazione della scheda. Inoltre, puoi visualizzare il motivo dell'errore in questa scheda.

Puoi anche scegliere il pulsante di attivazione della visualizzazione avanzata per visualizzare il percorso di trasferimento dei dati in uscita man mano che i dati passano attraverso lo stato selezionato. Ciò consente di identificare in che modo l'input è stato elaborato quando uno o più campi, ad esempio `InputPathParameters`,`ResultSelector`, e `OutputPathResultPath`, sono stati applicati ai dati.

Dettagli

Mostra informazioni, come il tipo di stato, lo stato di esecuzione e la durata dell'esecuzione.

Per `Task` gli stati che utilizzano una risorsa, ad esempio AWS Lambda, questa scheda fornisce collegamenti alla pagina di definizione della risorsa e alla pagina CloudWatch dei log di Amazon per la chiamata della risorsa. Mostra anche i valori, se specificati, per `Task` lo stato e i campi. `TimeoutSeconds` `HeartbeatSeconds`

Per `Map` gli stati, questa scheda mostra le informazioni relative al conteggio totale delle iterazioni di uno `Map` stato. Le iterazioni sono classificate come Non riuscite, Interrotte, Riuscite o. InProgress

Definizione

Mostra la definizione di Amazon States Language corrispondente allo stato selezionato.

Riprova

> **Note**
>
> Questa scheda viene visualizzata solo se hai definito un `Retry` campo nello `Parallel` stato `Task` o della tua macchina a stati.

Mostra i tentativi iniziali e successivi per uno stato selezionato nel tentativo di esecuzione originale. Per i tentativi falliti iniziali e per tutti i successivi tentativi falliti, scegli

accanto a Tipo per visualizzare il motivo dell'errore visualizzato in una casella a discesa. Se il nuovo tentativo ha esito positivo, puoi visualizzare l'output visualizzato in una casella a discesa.

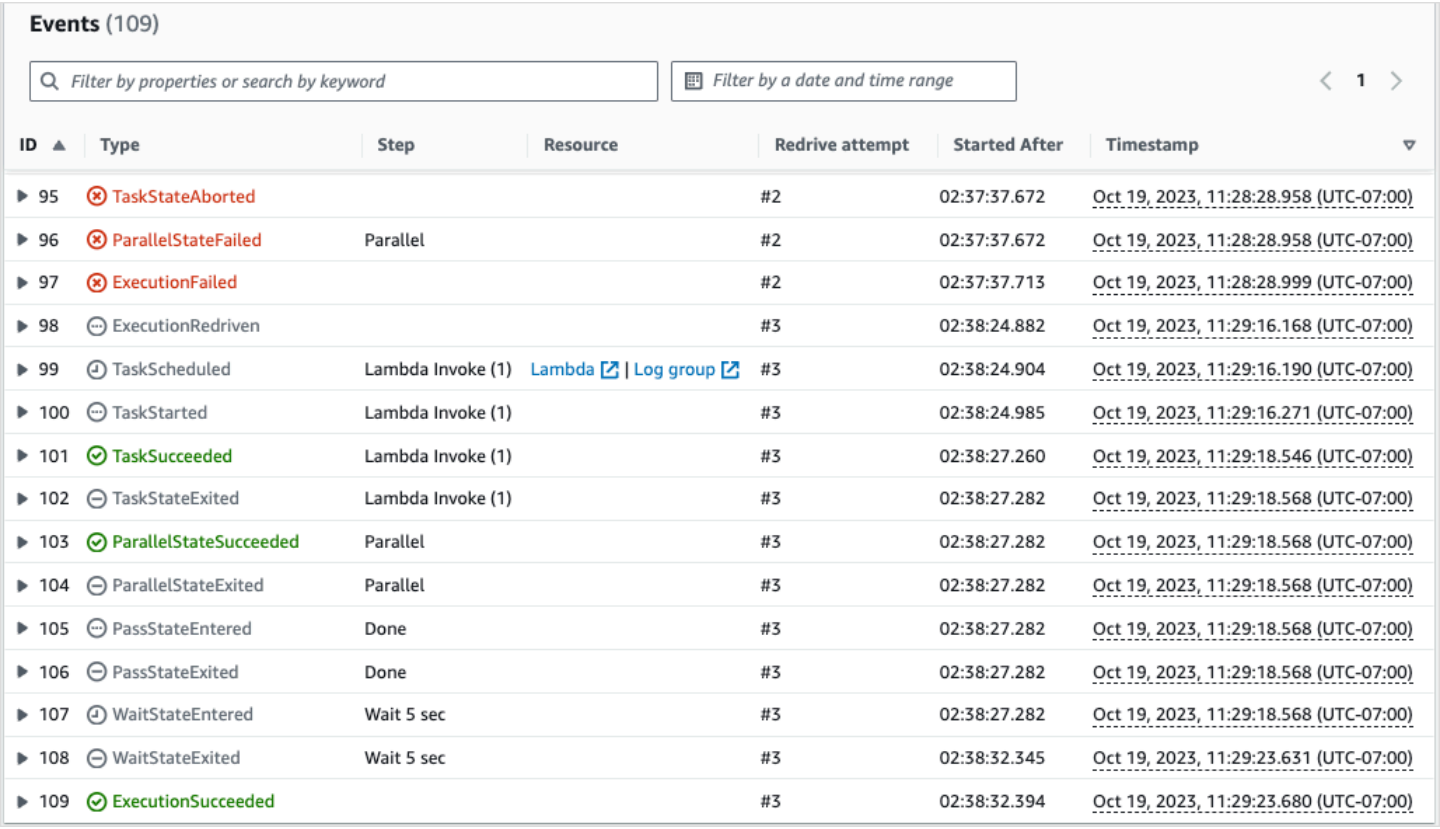
Se è in corso l'redrivenesecuzione, l'intestazione di questa scheda mostra il nome Riprova & redrives e mostra i dettagli del tentativo di nuovo tentativo per ciascuno di essi. redrive

Eventi

Mostra un elenco filtrato degli eventi associati allo stato selezionato in un'esecuzione. Le informazioni visualizzate in questa scheda sono un sottoinsieme della cronologia completa degli eventi di esecuzione visualizzata nella tabella Eventi.

Eventi

La tabella Eventi mostra la cronologia completa dell'esecuzione selezionata sotto forma di elenco di eventi che si estendono su più pagine. Ogni pagina contiene fino a 25 eventi. Questa sezione mostra anche il conteggio totale degli eventi, che può aiutarti a determinare se hai superato il numero massimo di 25.000 eventi nella cronologia degli eventi.



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)

Per impostazione predefinita, i risultati nella tabella Eventi vengono visualizzati in ordine crescente in base al timestamp degli eventi. È possibile modificare l'ordinamento della cronologia degli eventi di esecuzione in ordine decrescente facendo clic sull'intestazione della colonna Timestamp.

Nella tabella Eventi, ogni evento è codificato a colori per indicarne lo stato di esecuzione. Ad esempio, gli eventi che hanno avuto esito negativo vengono visualizzati in rosso. Per visualizzare ulteriori dettagli su un evento, scegli quello

accanto all'ID dell'evento. Una volta aperti, i dettagli dell'evento mostrano l'input, l'output e la chiamata delle risorse per l'evento.

Inoltre, nella tabella Eventi, è possibile applicare filtri per limitare i risultati della cronologia degli eventi di esecuzione visualizzati. È possibile scegliere proprietà come ID o Redrivetentativo. Per ulteriori informazioni, consulta Tutorial: Esame delle esecuzioni delle macchine a stati utilizzando la console Step Functions.

Tutorial: Esame delle esecuzioni delle macchine a stati utilizzando la console Step Functions

In questo tutorial, imparerai come controllare le informazioni sull'esecuzione visualizzate nella pagina Dettagli di esecuzione e visualizzare il motivo di un'esecuzione non riuscita. Quindi, imparerai come accedere a diverse iterazioni di un'esecuzione di `Map` stato. Infine, imparerai come configurare le colonne nella vista Tabella e applicare filtri adeguati per visualizzare solo le informazioni di tuo interesse.

In questo tutorial, creerai una macchina a stati di tipo Standard, che ottiene il prezzo di un set di frutta. Per fare ciò, la macchina a stati utilizza tre AWS Lambda funzioni che restituiscono un elenco casuale di quattro frutti, il prezzo di ogni frutto e il costo medio dei frutti. Le funzioni Lambda sono progettate per generare un errore se il prezzo dei frutti è inferiore o uguale a un valore di soglia.

> **Note**
>
> Sebbene la procedura seguente contenga istruzioni su come esaminare i dettagli dell'esecuzione di un flusso di lavoro Standard, è possibile esaminare anche i dettagli per le esecuzioni del flusso di lavoro Express. Per informazioni sulle differenze tra i dettagli di esecuzione per i tipi di flusso di lavoro Standard ed Express, vedereEsecuzioni Standard ed Express Workflow nella console.

Indice

- Fase 1: Creare e testare le funzioni Lambda richieste
- Fase 2: Creare ed eseguire la macchina a stati
- Fase 3: Visualizzare i dettagli di esecuzione della macchina a stati
- Fase 4: Esplora le diverse modalità di visualizzazione

Fase 1: Creare e testare le funzioni Lambda richieste

1. Apri la console Lambda ed esegui i passaggi da 1 a 4 della Fase 1: Creazione di una funzione Lambda sezione. Assicurati di assegnare un nome alla funzione Lambda. **`GetListOfFruits`**
2. Dopo aver creato la funzione Lambda, copia l'Amazon Resource Name (ARN) della funzione visualizzato nell'angolo in alto a destra della pagina. Per copiare l'ARN, fare clic su.

Di seguito è riportato un esempio di ARN, dove *`function-name`*è il nome della funzione Lambda (in questo caso,): `GetListOfFruits`

```
arn:aws:lambda:us-east-1:123456789012:function:function-name
```

3. Copia il codice seguente per la funzione Lambda nell'area Codice sorgente della GetListOfFruitspagina.

```
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. Scegli Deploy, quindi scegli Test, per distribuire le modifiche e vedere l'output della tua funzione Lambda.

5. Crea due funzioni Lambda aggiuntive, denominate **`GetFruitPrice`** e **`CalculateAverage`** rispettivamente, con i seguenti passaggi:

   a. Copia il codice seguente nell'area Codice sorgente della funzione 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. Copia il codice seguente nell'area Codice sorgente della funzione 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. Assicurati di copiare gli ARN di queste due funzioni Lambda, quindi distribuiscili e testali.

Fase 2: Creare ed eseguire la macchina a stati

Usa la console Step Functions per creare una macchina a stati che richiami le funzioni Lambda che hai creato nello Step 1. In questa macchina a stati, sono definiti tre `Map` stati. Ciascuno di questi `Map` stati contiene uno `Task` stato che richiama una delle tue funzioni Lambda. Inoltre, viene definito un `Retry` campo in ogni `Task` stato con un numero di tentativi di nuovo tentativo definiti per ogni stato. Se uno `Task` stato rileva un errore di runtime, viene eseguito nuovamente fino al numero di tentativi di nuovo tentativo definito per quello stato. `Task`

1. Apri la console Step Functions e scegli Scrivi il tuo flusso di lavoro nel codice.

> **Important**
>
> Assicurati che la tua macchina a stati utilizzi lo stesso AWS account e la stessa regione della funzione Lambda che hai creato in precedenza.

2. Per Tipo, mantieni la selezione predefinita di Standard.
3. Copia la seguente definizione di Amazon States Language e incollala in Definition. Assicurati di sostituire gli ARN mostrati con quelli delle funzioni Lambda che hai creato in precedenza.

```
{
    "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. Inserisci un nome per la tua macchina a stati. Mantieni le selezioni predefinite per le altre opzioni in questa pagina e scegli Crea macchina a stati.
5. Apri la pagina intitolata con il nome della tua macchina a stati. Eseguite i passaggi da 1 a 4 della Fase 4: Esegui la macchina a stati sezione, ma utilizzate i seguenti dati come input per l'esecuzione:

```
{
    "stores": [
      "Store A",
      "Store B",
      "Store C",
      "Store D"
    ]
}
```

Fase 3: Visualizzare i dettagli di esecuzione della macchina a stati

Nella pagina denominata con il tuo ID di esecuzione, puoi esaminare i risultati dell'esecuzione ed eseguire il debug di eventuali errori.

1. (Facoltativo) Scegliete tra le schede visualizzate nella pagina Dettagli di esecuzione per visualizzare le informazioni presenti in ciascuna di esse. Ad esempio, per visualizzare l'input della macchina a stati e il relativo output di esecuzione, scegliete Input e output di esecuzione nella sezione Riepilogo dell'esecuzione.
2. Se l'esecuzione della macchina a stati non è riuscita, scegli Causa o Mostra i dettagli del passaggio nel messaggio di errore. I dettagli sull'errore sono visualizzati nella sezione Dettagli del

passaggio. Notate che il passaggio che ha causato l'errore, ovvero uno `Task` stato denominato GetListofFruits, è evidenziato nella vista Grafico e nella vista Tabella.

> **Note**
>
> Poiché il GetListofFruitspassaggio è definito all'interno di uno `Map` stato e l'esecuzione non è riuscita, il passaggio **`Map`**Stato dello stato viene visualizzato come Non riuscito.

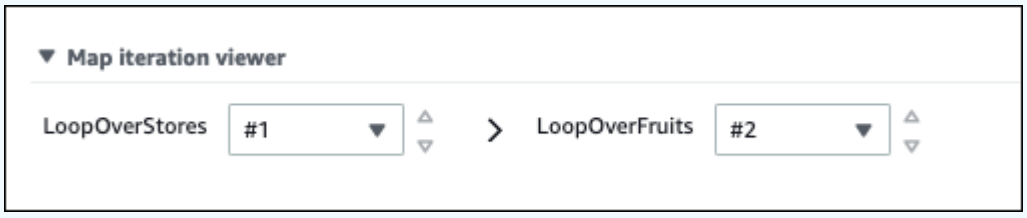
Fase 4: Esplora le diverse modalità di visualizzazione

È possibile scegliere una modalità preferita per visualizzare il flusso di lavoro della macchina a stati o la cronologia degli eventi di esecuzione. Alcune delle attività che è possibile eseguire in queste modalità di visualizzazione sono le seguenti:

Visualizzazione del grafico: consente di passare da un'iterazione di **`Map`** stato all'altra

Se lo stato della mappa ha cinque iterazioni e desideri visualizzare i dettagli di esecuzione per la terza e la quarta iterazione, procedi come segue:

1. Scegli lo `Map` stato per cui desideri visualizzare i dati di iterazione.
2. Dal Map Iteration Viewer, scegli l'iterazione che desideri visualizzare. Le iterazioni vengono contate da zero. Per scegliere la terza iterazione su cinque, scegli #2 dall'elenco a discesa accanto al nome dello stato della mappa.

> **Note**
>
> Se la macchina a stati contiene `Map` stati annidati, Step Functions visualizza le iterazioni `Map` dello stato principale e figlio come due elenchi a discesa separati:
>
> 
>

3. (Facoltativo) Se una o più iterazioni di `Map` stato non sono state eseguite o sono state interrotte in uno stato interrotto, è possibile visualizzare i dettagli sull'iterazione non riuscita. Per visualizzare questi dettagli, scegli i numeri di iterazione interessati in Non riuscito o Interrotto nell'elenco a discesa.

Visualizzazione a tabella: passa da un'iterazione di stato all'altra **`Map`**

Se lo stato della mappa ha cinque iterazioni e desideri visualizzare i dettagli di esecuzione per l'iterazione numero tre e quattro, procedi come segue:

1. Scegli lo `Map` stato per il quale desideri visualizzare i diversi dati di iterazione.
2. Nella visualizzazione ad albero delle iterazioni di `Map` stato, scegliete la riga per l'iterazione denominata #2 per l'iterazione numero tre. Allo stesso modo, scegli la riga denominata #3 per l'iterazione numero quattro.

Visualizzazione tabella: configura le colonne da visualizzare

Scegli

.

Quindi, nella finestra di dialogo Preferenze, scegliete le colonne che desiderate visualizzare in Seleziona colonne visibili.

Per impostazione predefinita, questa modalità visualizza le colonne Nome, Tipo, Stato, Risorsa e Started After.

Visualizzazione tabella: filtra i risultati

Limita la quantità di informazioni visualizzate applicando uno o più filtri in base a una proprietà, ad esempio Status, o a un intervallo di data e ora. Ad esempio, per visualizzare i passaggi che hanno avuto esito negativo nell'esecuzione, applica il seguente filtro:

1. Scegli Filtra per proprietà o cerca per parola chiave, quindi scegli Stato in Proprietà.
2. In Operatori, scegli Status =.
3. Scegliete Status = Failed.
4. (Facoltativo) Scegliete Cancella filtri per rimuovere i filtri applicati.

Visualizzazione degli eventi: filtra i risultati

Limita la quantità di informazioni visualizzate applicando uno o più filtri in base a una proprietà, ad esempio Type, o a un intervallo di data e ora. Ad esempio, per visualizzare i passaggi di `Task` stato che hanno avuto esito negativo nell'esecuzione, applica il seguente filtro:

1. Scegliete Filtra per proprietà o Cerca per parola chiave, quindi scegliete Tipo in Proprietà.

2. In Operatori, scegli Tipo =.
3. Scegliete Tipo = TaskFailed.
4. (Facoltativo) Scegliete Cancella filtri per rimuovere i filtri applicati.

Visualizzazione dell'evento: esamina i dettagli di un TaskFailedevento

Scegli l'icona accanto all'ID di un TaskFailedevento per esaminarne i dettagli, inclusi input, output e richiamo delle risorse che appaiono in una casella a discesa.

Redrivingesecuzioni

Puoi utilizzarlo redrive per riavviare le esecuzioni dei flussi di lavoro standard che non sono stati completati correttamente negli ultimi 14 giorni. Queste includono esecuzioni non riuscite, interrotte o scadute.

Quando si esegue redrive un'esecuzione, continua l'esecuzione non riuscita del passaggio non riuscito e utilizza lo stesso input. Step Functionsconserva i risultati e la cronologia di esecuzione dei passaggi riusciti e questi passaggi non vengono rieseguiti durante l'esecuzione. redrive Ad esempio, supponiamo che il flusso di lavoro contenga due stati: uno Pass stato seguito da uno stato. Attività Se l'esecuzione del flusso di lavoro non riesce nello stato dell'operazione e redrive l'esecuzione dell'operazione non riesce, l'esecuzione ripianifica e quindi riesegue lo stato dell'operazione.

Redrivenle esecuzioni utilizzano la stessa definizione della macchina a stati e lo stesso ARN di esecuzione utilizzato per il tentativo di esecuzione originale. Se il tentativo di esecuzione originale era associato a una versione, allo stesso alias o a entrambi, l'redrivenesecuzione è associata alla stessa versione, allo stesso alias o a entrambi. Anche se aggiorni l'alias in modo che punti a una versione diversa, l'redrivenesecuzione continua a utilizzare la versione associata al tentativo di esecuzione originale. Poiché redriven le esecuzioni utilizzano la stessa definizione di macchina a stati, è necessario avviare una nuova esecuzione se si aggiorna la definizione della macchina a stati.

Quando si esegue redrive un'esecuzione, il timeout a livello di macchina a stati, se definito, viene reimpostato su 0. Per ulteriori informazioni sul timeout a livello di macchina a stati, vedere. `TimeoutSeconds`

redrivesLe esecuzioni sono considerate come transizioni di stato. Per informazioni su come le transizioni di stato influiscono sulla fatturazione, consulta Step Functions Pricing.

Argomenti

- Redriveidoneità per esecuzioni non riuscite
- Redrivecomportamento dei singoli stati
- Autorizzazione IAM a redrive un'esecuzione
- Redrivingesecuzioni in console
- Redrivingesecuzioni tramite API
- redrivenEsame delle esecuzioni
- Riprova il comportamento delle esecuzioni redriven

Redriveidoneità per esecuzioni non riuscite

È possibile redrive eseguire esecuzioni se il tentativo di esecuzione originale soddisfa le seguenti condizioni:

- Hai iniziato l'esecuzione il 15 novembre 2023 o dopo tale data. Le esecuzioni iniziate prima di questa data non sono redrive idonee.
- Lo stato di esecuzione non `SUCCEEDED` lo è.
- L'esecuzione del flusso di lavoro non ha superato il redrivable periodo di 14 giorni. Redrivableil periodo si riferisce al periodo durante il quale è possibile redrive una determinata esecuzione. Questo periodo inizia dal giorno in cui una macchina a stati completa la sua esecuzione.
- L'esecuzione del flusso di lavoro non ha superato il tempo di apertura massimo di un anno. Per informazioni sulle quote di esecuzione delle macchine a stati, vedere. Quote relative alle esecuzioni di macchine a stati
- Il conteggio della cronologia degli eventi di esecuzione è inferiore a 24.999. Redrivenle esecuzioni aggiungono la cronologia degli eventi alla cronologia degli eventi esistente. Assicurati che l'esecuzione del flusso di lavoro contenga meno di 24.999 eventi per contenere l'evento della `ExecutionRedriven` cronologia e almeno un altro evento della cronologia.

Redrivecomportamento dei singoli stati

A seconda dello stato che ha avuto esito negativo nel flusso di lavoro, il redrive comportamento di tutti gli stati non riusciti varia. La tabella seguente descrive il redrive comportamento di tutti gli stati.

Nome dello stato	Redrivecomportamento di esecuzione
Pass	Se un passaggio precedente fallisce o la macchina a stati scade, lo stato Pass viene chiuso e non viene eseguito. redrive
Attività	Pianifica e riavvia lo stato dell'attività. Quando si redrive esegue un'esecuzione che riesegue uno stato Task, lo stato `TimeoutSeconds` for, se definito, viene reimpostato su 0. Per ulteriori informazioni sul timeout, consulta Task state.
Choice	Rivaluta le regole dello stato Choice.
Attendi	Se lo stato specifica `Timestamp` o `TimestampPath` fa riferimento a un timestamp passato, redrive provoca l'uscita dallo stato Wait e inserisce lo stato specificato nel campo. `Next`
Succeed	Non indica le esecuzioni automatiche che entrano redrive nello stato Succeed.
Fail	Rientra nello stato Fail e fallisce nuovamente.
Parallel	Riprogramma e redrives solo i rami che hanno avuto esito negativo o sono stati interrotti. Se lo stato ha avuto esito negativo a causa di un `States.DataLimitExceeded` errore, viene eseguito nuovamente lo stato Parallel, inclusi i rami che hanno avuto successo nel tentativo di esecuzione originale.
Stato della mappa in linea	Riprogramma e redrives solo le iterazioni fallite o interrotte.

Nome dello stato	Redrivecomportamento di esecuzione
	Se lo stato ha avuto esito negativo a causa di un `States.DataLimitExceeded` errore, viene eseguito nuovamente lo stato Inline Map, incluse le iterazioni che hanno avuto successo nel tentativo di esecuzione originale.
Stato della mappa distribuita	redrivesle esecuzioni non riuscite del workflow secondario in un Map Run. Per ulteriori informazioni, consulta RedrivingMappa corre. Se lo stato ha avuto esito negativo a causa di un `States.DataLimitExceeded` errore, lo stato della mappa distribuita viene eseguito nuovamente. Ciò include i flussi di lavoro secondari che hanno avuto successo nel tentativo di esecuzione originale.

Autorizzazione IAM a redrive un'esecuzione

Step Functions necessita dell'autorizzazione appropriata per redrive l'esecuzione. Il seguente esempio di policy IAM concede il minimo privilegio richiesto alla macchina a stati per l'esecuzioneredriving. Ricordati di sostituire il testo in *`corsivo con le informazioni specifiche`* della risorsa.

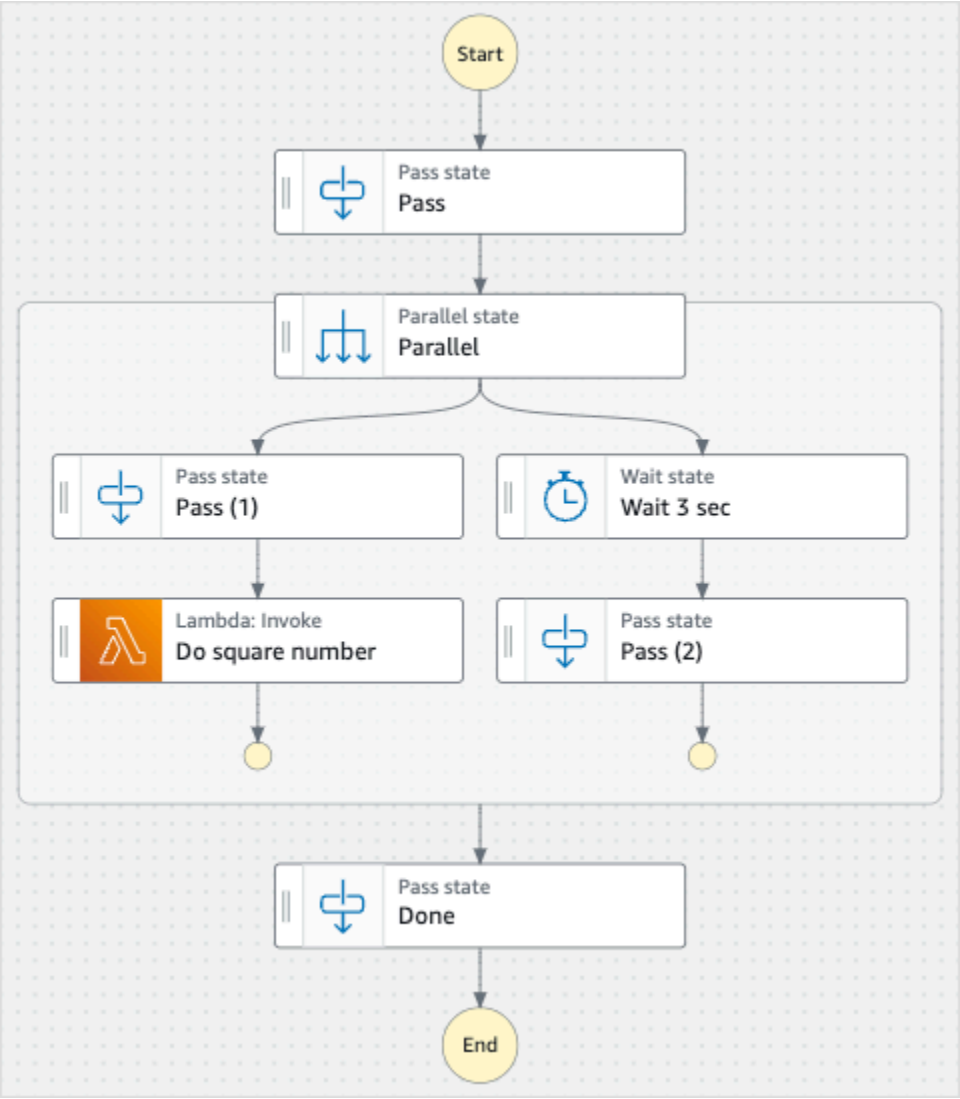
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "states:RedriveExecution"
            ],
            "Resource": "arn:aws:states:us-
east-2:123456789012:execution:myStateMachine:*"
        }
    ]
}
```

Per un esempio dell'autorizzazione necessaria per eseguire una Map Run, vedi. redrive Esempio di policy IAM per redriving una mappa distribuita

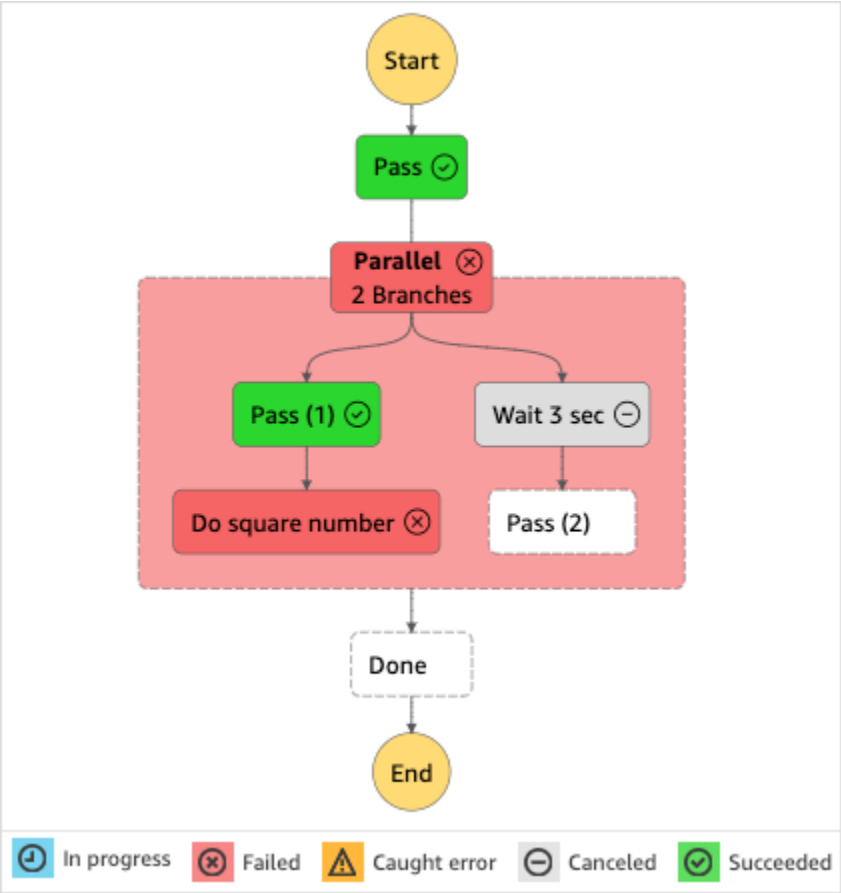
Redrivingesecuzioni in console

È possibile effettuare esecuzioni redrive idonee dalla Step Functions console.

Ad esempio, supponiamo che l'immagine seguente rappresenti il grafico del flusso di lavoro della macchina a stati.



Immagina di utilizzare questa macchina a stati. L'immagine seguente mostra il grafico dell'esecuzione della macchina a stati.



Come mostrato in questa immagine, il passaggio LambdaInvoke denominato Do square number all'interno dello stato Parallel ha restituito un errore. Ciò ha causato il fallimento dello stato Parallel. I rami la cui esecuzione era in corso o non avviata vengono interrotti e l'esecuzione della macchina a stati fallisce.

A redrive un'esecuzione dalla console

1. Apri la console Step Functions, quindi scegli una macchina a stati esistente la cui esecuzione non è riuscita.
2. Nella pagina dei dettagli della macchina a stati, in Esecuzioni, scegli un'istanza di esecuzione non riuscita.
3. Scegli Redrive.
4. Nella finestra Redrivedi dialogo, scegliete Redriveesecuzione.

> **Tip**
>
> Se ti trovi nella pagina Dettagli di esecuzione di un'esecuzione non riuscita, esegui una delle seguenti operazioni: redrive
>
> - Scegli Ripristina, quindi seleziona Redriveda un errore.
> - Scegli Azioni, quindi seleziona Redrive.

Si noti che redrive utilizza la stessa definizione di macchina a stati e lo stesso ARN. Continua a eseguire l'esecuzione dal passaggio che non è riuscito nel tentativo di esecuzione originale. In questo esempio, si tratta del passaggio Do square number e del ramo Wait 3 sec all'interno dello stato Parallel. Dopo aver riavviato l'esecuzione di questi passaggi non riusciti nello stato Parallel, redrive continuerà l'esecuzione per il passaggio Done.

5. Scegliete l'esecuzione per aprire la pagina Dettagli di esecuzione.

   In questa pagina è possibile visualizzare i risultati dell'redrivenesecuzione. Ad esempio, nella Riepilogo dell'esecuzione sezione è possibile visualizzare il Redriveconteggio, che rappresenta il numero di volte in cui è stata eseguita un'esecuzioneredriven. Nella sezione Eventi, è possibile visualizzare gli eventi di esecuzione redrive correlati aggiunti agli eventi del tentativo di esecuzione originale. Ad esempio, l'`ExecutionRedrive`nevento.

Redrivingesecuzioni tramite API

È possibile utilizzare l'API per le esecuzioni redrive idonee. RedriveExecution Questa API riavvia le esecuzioni non riuscite dei flussi di lavoro standard dal passaggio fallito, interrotto o scaduto.

In AWS Command Line Interface (AWS CLI), esegui il comando seguente per eseguire con successo una macchina a redrive stati. Ricordati di sostituire il testo in *`corsivo`* con le informazioni specifiche della risorsa.

```
aws stepfunctions redrive-execution --execution-arn arn:aws:states:us-
east-2:123456789012:execution:myStateMachine:foo
```

redrivenEsame delle esecuzioni

È possibile esaminare un'redrivenesecuzione nella console o utilizzando le API: and. GetExecutionHistoryDescribeExecution

Esamina le redriven esecuzioni sulla console

1. Apri la console Step Functions, quindi scegli una macchina a stati esistente per la quale hai redriven un'esecuzione.
2. Apri la pagina Dettagli di esecuzione.

   In questa pagina è possibile visualizzare i risultati dell'redrivenesecuzione. Ad esempio, nella Riepilogo dell'esecuzione sezione è possibile visualizzare il Redriveconteggio, che rappresenta il numero di volte in cui è stata eseguita un'esecuzioneredriven. Nella sezione Eventi, è possibile visualizzare gli eventi di esecuzione redrive correlati aggiunti agli eventi del tentativo di esecuzione originale. Ad esempio, l'`ExecutionRedrive`nevento.

Esamina redriven le esecuzioni utilizzando le API

Se hai redriven un'esecuzione su una macchina a stati, puoi utilizzare una delle seguenti API per visualizzare i dettagli sull'esecuzione. redriven Ricordati di sostituire il testo in *`corsivo`* con le informazioni specifiche della risorsa.

- GetExecutionHistory — Restituisce la cronologia dell'esecuzione specificata sotto forma di elenco di eventi. Questa API restituisce anche i dettagli sul redrive tentativo di esecuzione, se disponibili.

  In AWS CLI, esegui il comando seguente.

  ```
  aws stepfunctions get-execution-history --execution-arn arn:aws:states:us-
  east-2:123456789012:execution:myStateMachine:foo
  ```

- DescribeExecution — Fornisce informazioni sull'esecuzione di una macchina a stati. Può trattarsi della macchina a stati associata all'esecuzione, dell'input e dell'output dell'esecuzione, dei redrive dettagli di esecuzione, se disponibili, e dei metadati di esecuzione pertinenti.

  In AWS CLI, esegui il comando seguente.

  ```
  aws stepfunctions describe-execution --execution-arn arn:aws:states:us-
  east-2:123456789012:execution:myStateMachine:foo
  ```

Riprova il comportamento delle esecuzioni redriven

Se redriven l'esecuzione riesegue uno stato, o Inline Map AttivitàParallel, per il quale sono stati definiti nuovi tentativi, il conteggio dei tentativi per questi stati viene reimpostato su 0. Ciò consente il numero massimo di tentativi in corso. redrive Per un'redrivenesecuzione, è possibile tenere traccia dei singoli tentativi di nuovi tentativi di questi stati utilizzando la console.

Per esaminare i singoli tentativi di riprova nella console

1. Nella pagina Dettagli di esecuzione della console Step Functions, scegli uno stato su redrive cui è stato riprovato.
2. Scegliete la scheda Riprova e redrives.
3. Scegli il

   ▶

   successivo tentativo per visualizzarne i dettagli. Se il nuovo tentativo è riuscito, puoi visualizzare i risultati in Output, visualizzato in una casella a discesa.

L'immagine seguente mostra un esempio dei tentativi eseguiti per uno stato del tentativo di esecuzione originale e del redrives relativo tentativo. In questa immagine, vengono eseguiti tre tentativi nel tentativo originale e nel tentativo di redrive esecuzione. L'esecuzione riesce al quarto redrive tentativo e restituisce un risultato di 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   }
```

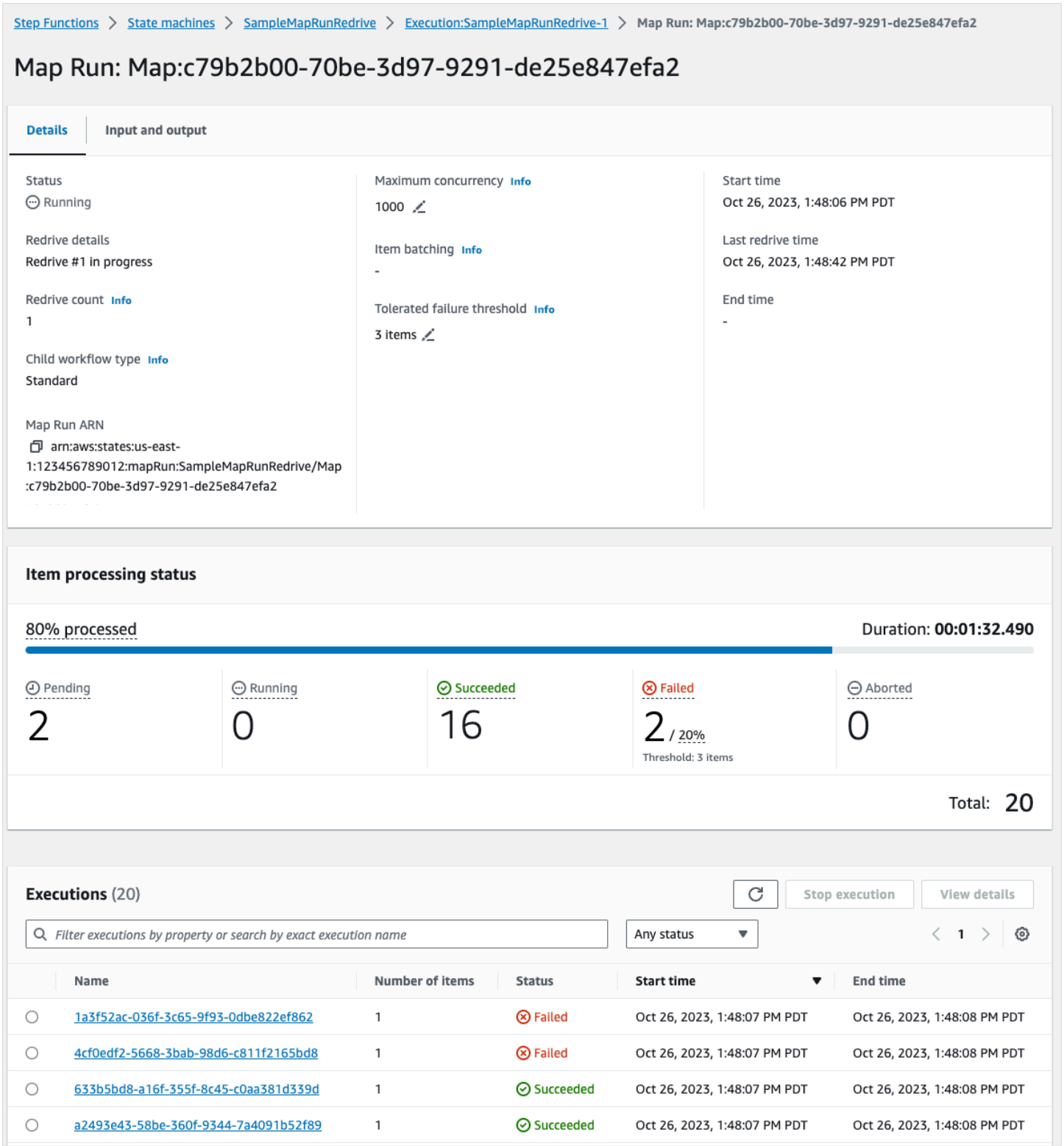
Formatted

Esame dell'esecuzione dello stato di una mappa distribuita in esecuzione

Quando si esegue uno `Map` stato in modalità Distribuita, Step Functions crea una risorsa Map Run. Un Map Run si riferisce a un insieme di esecuzioni di workflow secondarie avviate da uno stato di Distributed Map e alle impostazioni di runtime che controllano queste esecuzioni. Step Functions assegna un Amazon Resource Name (ARN) a Map Run. È possibile esaminare un Map Run nella console Step Functions. Puoi anche richiamare l'azione `DescribeMapRun` API. Un Map Run emette anche metriche per. CloudWatch

La console Step Functions fornisce una pagina Map Run Details che mostra tutte le informazioni relative all'esecuzione dello stato di una mappa distribuita. Ad esempio, è possibile visualizzare lo stato dell'esecuzione dello stato della mappa distribuita, l'ARN di Map Run e gli stati degli elementi elaborati nelle esecuzioni del flusso di lavoro secondario avviate dallo stato della mappa distribuita. È inoltre possibile visualizzare un elenco di tutte le esecuzioni dei flussi di lavoro secondari e accedere ai relativi dettagli. Inoltre, se il tuo Map Run lo era redriven, puoi vedere i redrive dettagli di Map Run nella Riepilogo dell'esecuzione di Map Run sezione. Ad esempio, L'ultima redrive volta. La console visualizza queste informazioni in un formato dashboard.

La pagina Map Run Details contiene le seguenti sezioni:



Indice

- Riepilogo dell'esecuzione di Map Run
- Messaggio di errore

- Stato di elaborazione dell'articolo
- Elenco delle esecuzioni
- RedrivingMappa corre
  - Redriveidoneità per i flussi di lavoro per bambini in Map Run
  - Comportamento di esecuzione del workflow secondario redrive
  - Scenari di input utilizzati su Map Run redrive
  - Autorizzazione IAM redrive a Map Run
  - RedrivingMap Run nella console
  - RedrivingMap Run utilizzando l'API

Riepilogo dell'esecuzione di Map Run

La sezione di riepilogo dell'esecuzione della mappa viene visualizzata nella parte superiore della pagina Dettagli di Map Run. Questa sezione fornisce una panoramica dei dettagli di esecuzione dello stato della mappa distribuita. Queste informazioni sono suddivise tra le seguenti schede:

Dettagli

Mostra informazioni, come lo stato di esecuzione dello stato della mappa distribuita, l'ARN di Map Run e il tipo di esecuzioni del flusso di lavoro secondario avviate dallo stato della mappa distribuita. È possibile visualizzare configurazioni aggiuntive, come la soglia di errore tollerata per Map Run e la concorrenza massima specificata per le esecuzioni di workflow secondarie. È inoltre possibile modificare queste configurazioni.

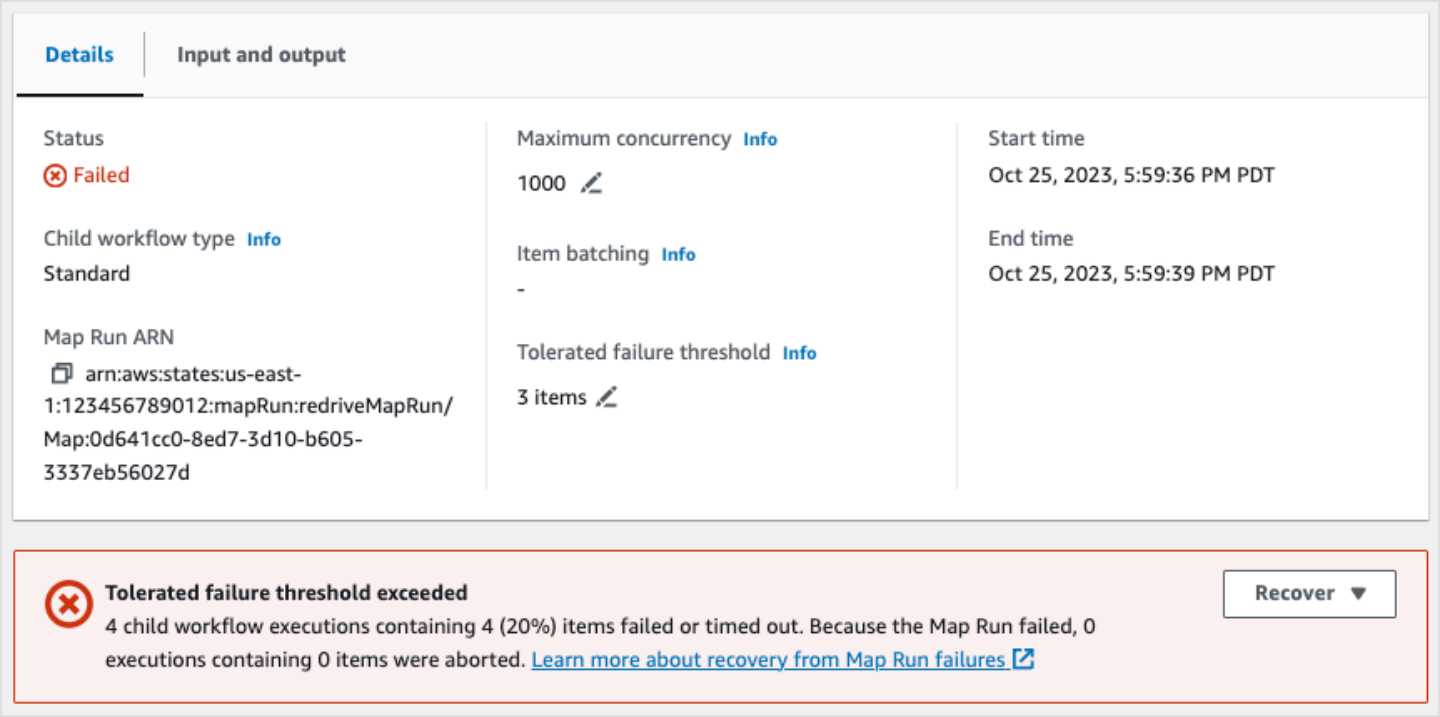
Ingresso e uscita

Mostra l'input ricevuto dallo stato della Mappa Distribuita e l'output corrispondente che genera. Ad esempio, è possibile visualizzare il set di dati di input e la relativa posizione e i filtri di input applicati ai singoli elementi di dati in quel set di dati. Se esporti l'output dell'esecuzione dello stato della mappa distribuita, questa scheda mostra il percorso del bucket Amazon S3 che contiene i risultati dell'esecuzione. Altrimenti, rimanda alla pagina dei dettagli di esecuzione del flusso di lavoro principale per visualizzare l'output dell'esecuzione.

Messaggio di errore

Se la Map Run non è riuscita, la pagina Dettagli Map Run visualizza un messaggio di errore con il motivo dell'errore.

Dal pulsante a discesa Recover di questo messaggio di errore, è possibile avviare redrive le esecuzioni non riuscite del workflow secondario avviate da questo Map Run o avviare una nuova esecuzione del workflow principale. Per ulteriori informazioni, consulta RedrivingMappa corre.



Stato di elaborazione dell'articolo

La sezione Stato di elaborazione degli articoli mostra lo stato degli elementi elaborati in una Map Run. Ad esempio, Pending indica che l'esecuzione di un workflow secondario non ha ancora avviato l'elaborazione dell'elemento.

Gli stati degli elementi dipendono dallo stato delle esecuzioni del flusso di lavoro secondario che elaborano gli elementi. Se l'esecuzione di un flusso di lavoro secondario non riesce, scade o se un utente annulla l'esecuzione, Step Functions non riceve alcuna informazione sul risultato dell'elaborazione degli elementi all'interno di tale esecuzione del flusso di lavoro secondario. Tutti gli elementi elaborati da tale esecuzione condividono lo stato di esecuzione del workflow secondario.

Ad esempio, supponiamo di voler elaborare 100 elementi in due esecuzioni di workflow secondarie, in cui ogni esecuzione elabora un batch di 50 elementi. Se una delle esecuzioni fallisce e l'altra riesce, avrai 50 elementi riusciti e 50 elementi non riusciti.

La tabella seguente illustra i tipi di stati di elaborazione disponibili per tutti gli elementi:

Stato	Descrizione
Pending (In attesa)	Indica un elemento che l'esecuzione del workflow secondario non ha avviato l'elaborazione. Se una Map Run si interrompe, fallisce o un utente annulla l'esecuzione prima dell'inizio dell'elaborazione di un elemento, l'elemento rimane nello stato In sospeso. Ad esempio, se una Map Run non riesce con 10 elementi in sospeso da elaborare, questi 10 elementi rimangono nello stato In sospeso.
In esecuzione	Indica un elemento attualmente in fase di elaborazione da parte dell'esecuzione del workflow secondario.
Riuscito	Indica che l'esecuzione del workflow secondario ha elaborato correttamente l'elemento. Un'esecuzione corretta del flusso di lavoro secondario non può avere alcun elemento non riuscito. Se un elemento del set di dati fallisce durante l'esecuzione, l'intera esecuzione del workflow secondario ha esito negativo.
Failed (Non riuscito)	Indica che l'esecuzione del workflow secondario non è riuscita a elaborare l'elemento o che l'esecuzione è scaduta. Se l'esecuzione di un elemento elaborato da un workflow secondario non riesce, l'intera esecuzione del workflow secondario ha esito negativo. Ad esempio, si consideri l'esecuzione di un flusso di lavoro secondario che ha elaborato 1000 elementi. Se un elemento del set di dati si guasta durante l'esecuzione, Step Functions

Stato	Descrizione
	considera l'intera esecuzione del workflow secondario come fallita. Quando si esegue redriveuna Map Run, il numero di elementi con questo stato viene reimpostato su 0.
Interrotto	Indica che l'esecuzione del workflow secondari o ha iniziato l'elaborazione dell'elemento, ma l'utente ha annullato l'esecuzione o Step Functions ha interrotto l'esecuzione perché l'esecuzione della mappa non è riuscita. Ad esempio, si consideri l'esecuzione di un workflow secondario in esecuzione che elabora 50 elementi. Se l'esecuzione della mappa si interrompe a causa di un errore o perché un utente ha annullato l'esecuzione, l'esecuzione del flusso di lavoro secondario e lo stato di tutti i 50 elementi diventano Interrotti. Se si utilizza un'esecuzione di workflow secondaria di tipo Express, non è possibile interrompere l'esecuzione. Quando si esegue redriveun Map Run che avvia esecuzioni di workflow secondarie di tipo Express, il numero di elementi con questo stato viene reimpostato su 0. Questo perché i flussi di lavoro secondari di Express vengono riavviati utilizzando l'azione StartExecutionAPI anziché esserlo. redriven

Elenco delle esecuzioni

La sezione Esecuzioni elenca tutte le esecuzioni del workflow secondario per una specifica Map Run. Utilizza il campo Cerca per nome esatto dell'esecuzione per cercare un'esecuzione specifica del workflow secondario. Puoi anche utilizzare il menu a discesa Qualsiasi stato per filtrare le cronologie di esecuzione dei flussi di lavoro secondari in base al loro stato. Per visualizzare i dettagli su un'esecuzione specifica, seleziona un'esecuzione del flusso di lavoro secondario dall'elenco e scegli il pulsante Visualizza dettagli per aprire la relativa pagina dei dettagli di esecuzione.

Important

La politica di conservazione per le esecuzioni di flussi di lavoro secondari è di 90 giorni. Le esecuzioni di flussi di lavoro secondari completate che sono precedenti a questo periodo di conservazione non vengono visualizzate nella tabella Esecuzioni. Ciò è vero anche se lo stato della Mappa distribuita o il flusso di lavoro principale continuano a funzionare più a lungo del periodo di conservazione. Puoi visualizzare i dettagli di esecuzione, inclusi i risultati, di queste esecuzioni di workflow secondarie se esporti l'output dello stato di Distributed Map in un bucket Amazon S3 utilizzando. `ResultWriter`

Tip

Scegli il pulsante di aggiornamento

per visualizzare l'elenco più aggiornato di tutte le esecuzioni di flussi di lavoro secondari.

RedrivingMappa corre

È possibile riavviare le esecuzioni di workflow secondarie non riuscite in una mappa eseguita dal redrivingflusso di lavoro principale. Un flusso di lavoro redriven principale con redrives tutti gli stati non riusciti, inclusa Distributed Map. Un flusso di lavoro principale rigenera gli stati non riusciti se non c'è alcun `<stateType>Exited` evento corrispondente all'`<stateType>Entered`evento per uno stato in cui il flusso di lavoro principale ha completato la sua esecuzione. Ad esempio, se la cronologia degli eventi non contiene l'evento relativo a un `MapStateExited` `MapStateEntered` evento, è possibile utilizzare redrive il flusso di lavoro principale per eseguire redrive tutte le esecuzioni di workflow secondarie non riuscite in Map Run.

Un Map Run non viene avviato o ha esito negativo nel tentativo di esecuzione originale quando la macchina a stati non dispone dell'autorizzazione richiesta per accedere a ItemReaderResultWriter, o a entrambi. Se Map Run non è stato avviato nel tentativo di esecuzione originale del workflow principale, redriving il workflow principale avvia Map Run per la prima volta. Per risolvere questo problema, aggiungi le autorizzazioni richieste al tuo ruolo di macchina a stati e quindi al flusso di lavoro redrive principale. Se utilizzi redrive il flusso di lavoro principale senza aggiungere le autorizzazioni richieste, tenta di avviare una nuova esecuzione di Map Run che avrà nuovamente esito negativo. Per informazioni sulle autorizzazioni di cui potresti aver bisogno, consulta. Politiche IAM per l'utilizzo dello stato della mappa distribuita

Argomenti

- Redriveidoneità per i flussi di lavoro per bambini in Map Run
- Comportamento di esecuzione del workflow secondario redrive
- Scenari di input utilizzati su Map Run redrive
- Autorizzazione IAM redrive a Map Run
- RedrivingMap Run nella console
- RedrivingMap Run utilizzando l'API

Redriveidoneità per i flussi di lavoro per bambini in Map Run

È possibile annullare redrive le esecuzioni di workflow secondarie non riuscite in una Map Run se sono soddisfatte le seguenti condizioni:

- Hai iniziato l'esecuzione del flusso di lavoro principale il o dopo il 15 novembre 2023. Le esecuzioni iniziate prima di questa data non sono redrive idonee.
- Non hai superato il limite massimo redrives di 1000 per un determinato Map Run. Se hai superato questo limite, riceverai l'`States.Runtime`errore.
- Il flusso di lavoro principale èredrivable. Se il flusso di lavoro principale non lo èredrivable, non è possibile eseguire redrive il flusso di lavoro secondario in un Map Run. Per ulteriori informazioni sull'redriveidoneità di un flusso di lavoro, consulta. Redriveidoneità per esecuzioni non riuscite
- Le esecuzioni secondarie del flusso di lavoro di tipo Standard in Map Run non hanno superato il limite di 25.000 eventi di esecuzione cronologici. Le esecuzioni di workflow secondarie che hanno superato il limite della cronologia degli eventi vengono conteggiate ai fini della soglia di errore tollerata e considerate non riuscite. Per ulteriori informazioni sull'redriveidoneità di un'esecuzione, vedere. Redriveidoneità per esecuzioni non riuscite

Viene avviata una nuova Map Run e la Map Run esistente non lo è redriven nei seguenti casi, anche se la Map Run non è riuscita nel tentativo di esecuzione originale:

- Map Run non è riuscito a causa dell'`States.DataLimitExceeded`errore.
- Map Run non è riuscito a causa dell'errore di interpolazione dei dati JSON,. `States.Runtime` Ad esempio, hai selezionato un nodo JSON inesistente in. OutputPath

Una Map Run può continuare a funzionare anche dopo l'arresto o il timeout del flusso di lavoro principale. In questi scenari, redrive ciò non avviene immediatamente:

- Map Run potrebbe ancora annullare le esecuzioni di workflow secondari in corso di tipo Standard o attendere che le esecuzioni di workflow secondarie di tipo Express completino le proprie esecuzioni.
- Map Run potrebbe ancora scrivere risultati suResultWriter, se l'hai configurato per esportare i risultati.

In questi casi, Map Run in esecuzione completa le sue operazioni prima di tentare di farlo. redrive

Comportamento di esecuzione del workflow secondario redrive

Le esecuzioni dei flussi di lavoro redriven secondari in un Map Run presentano il comportamento descritto nella tabella seguente.

Flusso di lavoro secondario Express	Flusso di lavoro standard per bambini
Tutte le esecuzioni di workflow secondarie non riuscite o scadute nel tentativo di esecuzione originale vengono avviate utilizzando l'azione StartExecutionAPI. Il primo stato in ingresso ItemProcessor viene eseguito per primo.	Tutte le esecuzioni di workflow secondarie non riuscite, scadute o annullate nel tentativo di esecuzione originale redriven utilizzano l'azione API. RedriveExecution Questi flussi di lavoro secondari provengono redriven dall'ultimo stato in cui la loro esecuzione ItemProcessor non è riuscita.
Le esecuzioni non hanno sempre successo. redriven Questo perché le esecuzioni del flusso di lavoro secondario di Express vengono	Le esecuzioni dei flussi di lavoro secondari standard non hanno sempre esito positivo. redriven Se l'esecuzione non lo èredrivable, non verrà tentata di nuovo. L'ultimo errore

Flusso di lavoro secondario Express	Flusso di lavoro standard per bambini
sempre avviate come una nuova esecuzione utilizzando l' StartExecution azione API.	o output dell'esecuzione è permanente. Ciò è possibile quando un'esecuzione supera i 25.000 eventi cronologici o il redrivable periodo di 14 giorni è scaduto. L'esecuzione del flusso di lavoro secondario standard potrebbe non avvenire redrivable se l'esecuzione del flusso di lavoro principale si è conclusa entro 14 giorni, ma l'esecuzione del flusso di lavoro secondario si è conclusa prima di 14 giorni.
Le esecuzioni di workflow secondarie Express utilizzano lo stesso ARN di esecuzione del tentativo di esecuzione originale, ma non è possibile identificarne distintamente l'individuo. redrives	Le esecuzioni di workflow secondarie standard utilizzano lo stesso ARN di esecuzione del tentativo di esecuzione originale. È possibile identificare distintamente l'individuo redrives nella console e utilizzando API, come e. GetExecutionHistoryDescribeExecution Per ulteriori informazioni, consulta redrivenEsame delle esecuzioni.

Se hai redriven una Map Run che ha raggiunto il limite di concorrenza, le esecuzioni del flusso di lavoro secondario in quel Map Run passano allo stato in sospeso. Lo stato di esecuzione di Map Run passa anche allo stato In sospeso. redrive Fino a quando il limite di concorrenza specificato non consente l'esecuzione di più esecuzioni di workflow secondari, l'esecuzione rimane nello stato In sospeso. redrive

Ad esempio, supponiamo che il limite di concorrenza della mappa distribuita nel flusso di lavoro sia 3000 e che il numero di flussi di lavoro secondari da eseguire nuovamente sia 6000. Ciò fa sì che 3000 flussi di lavoro secondari vengano eseguiti in parallelo mentre i restanti 3000 flussi di lavoro rimangano nello stato di redrive in sospeso. Una volta completata l'esecuzione del primo batch di 3000 flussi di lavoro secondari, vengono eseguiti i restanti 3000 flussi di lavoro secondari.

Quando una Map Run ha completato la sua esecuzione o viene interrotta, il numero di esecuzioni di workflow secondarie redrive nello stato In sospeso viene reimpostato su 0.

Scenari di input utilizzati su Map Run redrive

A seconda del modo in cui hai fornito l'input alla Mappa Distribuita nel tentativo di esecuzione originale, redriven Map Run utilizzerà l'input come descritto nella tabella seguente.

Inserimento nel tentativo di esecuzione originale	Input utilizzato su Map Run redrive
Input passato da uno stato precedente o dall'input di esecuzione.	redrivenMap Run utilizza lo stesso input.
L'input è passato utilizzando ItemReader e Map Run non ha avviato le esecuzioni del workflow secondario perché è vera una delle seguenti condizioni: • Map Run non è riuscito con l'`States.ItemReaderFailed` errore. • Map Run non è riuscito con l'`States.ResultWriterFailed` errore. • L'esecuzione del workflow principale è stata interrotta o annullata prima dell'avvio di Map Run.	redrivenMap Run utilizza l'input nel bucket Amazon S3.
Input passato utilizzando. ItemReader Map Run non è riuscito dopo l'avvio o il tentativo di avviare esecuzioni secondarie di workflow.	redrivenMap Run utilizza lo stesso input fornito nel tentativo di esecuzione originale.

Autorizzazione IAM redrive a Map Run

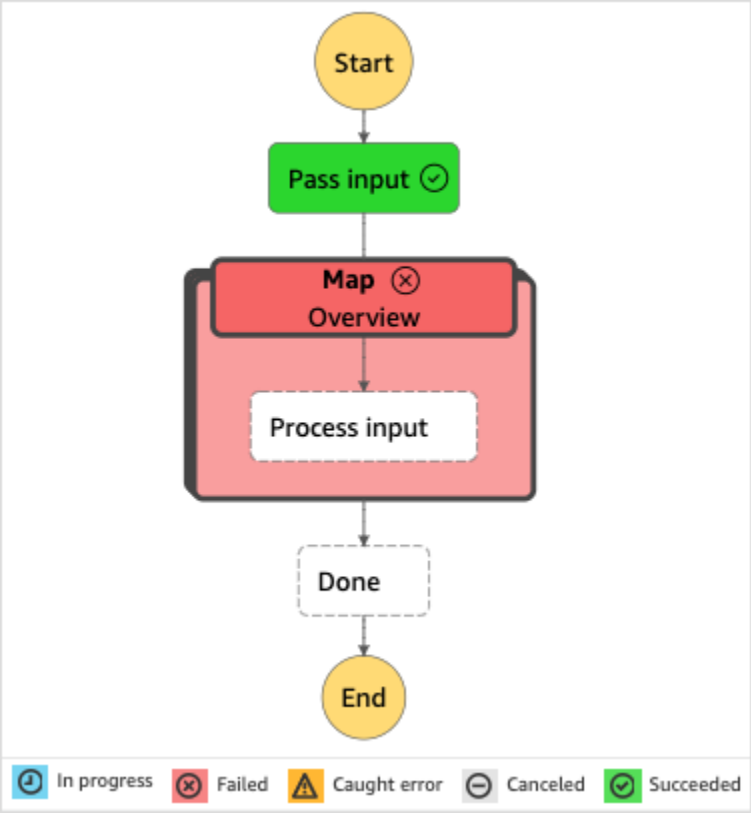
Step Functions necessita dell'autorizzazione appropriata per eseguire redrive una Map Run. Il seguente esempio di policy IAM concede il minimo privilegio richiesto alla macchina a stati per redriving una Map Run. Ricordati di sostituire il testo in *`corsivo con le informazioni specifiche`* della risorsa.

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

```
    {
      "Effect": "Allow",
      "Action": [
        "states:RedriveExecution"
      ],
      "Resource": "arn:aws:states:us-
east-2:123456789012:execution:myStateMachine/myMapRunLabel:*"
    }
  ]
}
```

RedrivingMap Run nella console

L'immagine seguente mostra il grafico di esecuzione di una macchina a stati che contiene una mappa distribuita. Questa esecuzione non è riuscita perché l'esecuzione della mappa non è riuscita. Per redrive Map Run, è necessario redrive il flusso di lavoro principale.



A redrive un Map Run dalla console

1. Apri la console Step Functions, quindi scegli una macchina a stati esistente che contenga una mappa distribuita la cui esecuzione non è riuscita.

2. Nella pagina dei dettagli della macchina a stati, in Esecuzioni, scegli un'istanza di esecuzione non riuscita di questa macchina a stati.
3. Scegli Redrive.
4. Nella finestra Redrivedi dialogo, scegliete Redriveesecuzione.

> **Tip**
>
> È inoltre possibile eseguire redrive un Map Run dalla pagina Dettagli di esecuzione o Dettagli di esecuzione della mappa.
> Se ti trovi nella pagina Dettagli di esecuzione, esegui una delle seguenti operazioni: redrive
>
> - Scegli Ripristina, quindi seleziona Redriveda errore.
> - Scegli Azioni, quindi seleziona Redrive.
>
> Se ti trovi nella pagina Map Run Details, scegli Recupera, quindi seleziona RedriveExtra Failure.

Si noti che redrive utilizza la stessa definizione di macchina a stati e lo stesso ARN. Continua a eseguire l'esecuzione dal passaggio che non è riuscito nel tentativo di esecuzione originale. In questo esempio, si tratta del passaggio Distributed Map denominato Map e del passaggio di input Process al suo interno. Dopo aver riavviato le esecuzioni non riuscite del workflow secondario di Map Run, redrive continuerà l'esecuzione per il passaggio Done.

5. Dalla pagina Dettagli di esecuzione, scegli Map Run per vedere i dettagli di Map Run. redriven

   In questa pagina puoi visualizzare i risultati dell'redrivenesecuzione. Ad esempio, nella Riepilogo dell'esecuzione di Map Run sezione, puoi vedere il Redriveconteggio, che rappresenta il numero di volte in cui è stata eseguita la Map Runredriven. Nella sezione Eventi, è possibile visualizzare gli eventi di esecuzione redrive correlati aggiunti agli eventi del tentativo di esecuzione originale. Ad esempio, l'`MapRunRedrive`nevento.

Dopo aver eseguito redriven una Map Run, puoi esaminarne redrive i dettagli nella console o utilizzare le azioni GetExecutionHistorye DescribeExecutionAPI. Per ulteriori informazioni sull'esame di un'redrivenesecuzione, consultaredrivenEsame delle esecuzioni.

RedrivingMap Run utilizzando l'API

Puoi creare redrive un Map Run idoneo utilizzando l'RedriveExecutionAPI nel flusso di lavoro principale. Questa API riavvia le esecuzioni di workflow secondarie non riuscite in un Map Run.

In AWS Command Line Interface (AWS CLI), esegui il seguente comando per redrive un'esecuzione non riuscita di una macchina a stati. Ricordati di sostituire il testo in *corsivo* con le informazioni specifiche della risorsa.

```
aws stepfunctions redrive-execution --execution-arn arn:aws:states:us-
east-2:123456789012:execution:myStateMachine:foo
```

Dopo aver eseguito redriven una Map Run, puoi esaminarne redrive i dettagli nella console o utilizzando l'azione API. DescribeMapRun Per esaminare i redrive dettagli delle esecuzioni dei flussi di lavoro standard in un Map Run, puoi utilizzare l'azione GetExecutionHistoryo DescribeExecutionAPI. Per ulteriori informazioni sull'esame di un'redrivenesecuzione, vedere. the section called “redrivenEsame delle esecuzioni”

Puoi esaminare redrive i dettagli delle esecuzioni del flusso di lavoro Express in un Map Run sulla console Step Functions se hai abilitato la registrazione sul flusso di lavoro principale. Per ulteriori informazioni, consulta Registrazione utilizzandoCloudWatchRegistri.

Gestione degli errori in Step Functions

Tutti gli stati, tranne `Wait` gli stati `Pass` e, possono presentare errori di runtime. Gli errori possono verificarsi per vari motivi, ad esempio i seguenti esempi:

- Problemi relativi alla definizione della macchina a stati (ad esempio, assenza di regola corrispondente in uno stato `Choice`)
- Esito negativo dei task (ad esempio un'eccezione in una funzione AWS Lambda)
- Problemi transitori (ad esempio eventi di partizione della rete)

Per impostazione predefinita, quando uno stato riporta un errore, AWS Step Functions causa il totale insuccesso dell'esecuzione.

> **Tip**
>
> Per implementare un esempio di flusso di lavoro che include la gestione degli errori nel vostro ambienteAccount AWS, consultate il Modulo 8 - Gestione degli errori di The AWS Step Functions Workshop.

Argomenti

- Nomi degli errori
- Riprovare dopo un errore
- Stati di fallback
- Esempi di macchine a stati che utilizzano Retry e Catch

Nomi degli errori

Step Functions identifica gli errori in Amazon States Language utilizzando stringhe con distinzione tra maiuscole e minuscole, note come nomi di errore. Amazon States Language definisce un set di stringhe integrate che denominano errori noti, tutti a partire dal `States.` prefisso.

`States.ALL`

Un carattere jolly che corrisponde a qualsiasi nome di errore noto.

> **Note**
>
> Questo tipo di errore non è in grado di catturare il tipo di errore del `States.DataLimitExceeded` terminale e i tipi di errore di runtime. Per ulteriori informazioni su questi tipi di errore, consulta `States.DataLimitExceeded`e `States.Runtime`.

`States.DataLimitExceeded`

Step Functions riporta un'`States.DataLimitExceeded`eccezione nelle seguenti condizioni:

- Quando l'uscita di un connettore supera la quota di dimensione del payload.
- Quando l'uscita di uno stato è maggiore della quota di dimensione del payload.

- Quando, dopo l'`Parameters`elaborazione, l'input di uno stato è maggiore della quota di dimensione del payload.

Per ulteriori informazioni sulle quote, vedere. Quote

> **Note**
>
> Si tratta di un errore del terminale che non può essere rilevato dal tipo di `States.ALL` errore.

`States.ExceedToleratedFailureThreshold`

Uno `Map` stato non è riuscito perché il numero di elementi non riusciti ha superato la soglia specificata nella definizione della macchina a stati. Per ulteriori informazioni, consulta Soglia di errore tollerata per lo stato della mappa distribuita.

`States.HeartbeatTimeout`

Uno `Task` stato non è riuscito a inviare un battito cardiaco per un periodo più lungo del `HeartbeatSeconds` valore.

> **Note**
>
> Questo errore è disponibile solo all'interno dei `Retry` campi `Catch` e.

`States.IntrinsicFailure`

Un tentativo di richiamare una funzione intrinseca all'interno di un modello di payload non è riuscito.

`States.ItemReaderFailed`

Uno `Map` stato non è riuscito perché non è stato in grado di leggere la fonte dell'elemento specificata nel campo. `ItemReader` Per ulteriori informazioni, consulta `ItemReader`.

`States.NoChoiceMatched`

Uno `Choice` stato non è riuscito a far corrispondere l'input alle condizioni definite nella regola di scelta e non è stata specificata una transizione predefinita.

`States.ParameterPathFailure`

Un tentativo di sostituire un campo, all'interno del `Parameters` campo di uno stato, il cui nome termina `.$` con l'utilizzo di un percorso fallisce.

`States.Permissions`

Uno `Task` stato non è riuscito perché non disponeva di privilegi sufficienti per eseguire il codice specificato.

`States.ResultPathMatchFailure`

Step Functions non è riuscito ad applicare il `ResultPath` campo di uno stato all'input ricevuto dallo stato.

`States.ResultWriterFailed`

Uno `Map` stato non è riuscito perché non è stato in grado di scrivere risultati nella destinazione specificata nel `ResultWriter` campo. Per ulteriori informazioni, consulta `ResultWriter`.

`States.Runtime`

Un'esecuzione non è riuscita a causa di un'eccezione che non è stata in grado di elaborare. Spesso questo è causato da errori in fase di esecuzione, come il tentativo di applicare `InputPath` o `OutputPath` su un payload JSON null. Un `States.Runtime` errore non è recuperabile e causerà sempre il fallimento dell'esecuzione. Un nuovo tentativo o un catch on `States.ALL` non rileveranno `States.Runtime` errori.

`States.TaskFailed`

Uno stato `Task` ha avuto esito negativo durante l'esecuzione. Quando viene utilizzato in un retry o catch, `States.TaskFailed` funge da jolly che corrisponde a qualsiasi nome di errore noto ad eccezione di. `States.Timeout`

`States.Timeout`

Uno stato `Task` è rimasto in esecuzione più a lungo del valore di `TimeoutSeconds` oppure non è riuscito a inviare un heartbeat per un periodo di durata superiore al valore `HeartbeatSeconds`.

Inoltre, se una macchina a stati funziona più a lungo del `TimeoutSeconds` valore specificato, l'esecuzione fallisce con un `States.Timeout` errore.

Gli stati possono riportare errori con altri nomi. Tuttavia, questi nomi di errore non possono iniziare con il `States.` prefisso.

Come best practice, assicurati che il codice di produzione sia in grado di gestire le eccezioni del servizio AWS Lambda (`Lambda.ServiceException` e `Lambda.SdkClientException`). Per ulteriori informazioni, consulta Gestire le eccezioni del servizio Lambda.

> **Note**
>
> Gli errori non gestiti in Lambda vengono riportati come `Lambda.Unknown` nell'output degli errori. Questi includono out-of-memory errori e timeout delle funzioni. Puoi abbinare o `States.TaskFailed` gestire questi errori. `Lambda.Unknown` `States.ALL` Quando Lambda raggiunge il numero massimo di chiamate, l'errore è. `Lambda.TooManyRequestsException` Per ulteriori informazioni sugli errori della funzione Lambda, consulta Gestione degli errori e tentativi automatici nella Developer Guide. AWS Lambda

Riprovare dopo un errore

`TaskParallel`, e `Map` gli stati possono avere un campo denominato`Retry`, il cui valore deve essere una matrice di oggetti noti come retrier. Un singolo retrier rappresenta un determinato numero di tentativi, di solito a intervalli di tempo crescenti.

Quando uno di questi stati segnala un errore ed è presente un `Retry` campo, Step Functions analizza i retrier nell'ordine elencato nell'array. Quando il nome dell'errore compare nel valore del `ErrorEquals` campo di un retrier, la macchina a stati esegue nuovi tentativi come definito nel campo. `Retry`

Se redriven l'esecuzione esegue nuovamente uno statoAttività, o Inline Map Parallel, per il quale sono stati definiti nuovi tentativi, il conteggio dei tentativi per questi stati viene ripristinato a 0 per consentire il numero massimo di tentativi. redrive Per un'redrivenesecuzione, è possibile tenere traccia dei singoli tentativi di nuovi tentativi di questi stati utilizzando la console. Per ulteriori informazioni, consulta Riprova il comportamento delle esecuzioni redriven in Redrivingesecuzioni.

Un retrier contiene i seguenti campi:

> **Note**
>
> I nuovi tentativi vengono trattati come transizioni di stato. Per informazioni su come le transizioni di stato influiscono sulla fatturazione, consulta Step Functions Pricing.

`ErrorEquals` (Obbligatorio)

Un array non vuoto di stringhe che corrispondono a nomi di errore. Quando uno stato segnala un errore, Step Functions analizza i retrier. Quando nell'array compare il nome dell'errore, implementa la policy per i nuovi tentativi descritta nel retrier.

`IntervalSeconds` (facoltativo).

Un numero intero positivo che rappresenta il numero di secondi prima del primo tentativo (`1`per impostazione predefinita). `IntervalSeconds`ha un valore massimo di 99999999.

`MaxAttempts` (facoltativo).

Un numero intero positivo che rappresenta il numero massimo di tentativi (`3` per impostazione predefinita). Se l'errore si ripete più volte di quanto specificato, i tentativi cessano e viene ripristinata la gestione normale degli errori. Il valore di `0` specifica che l'errore non viene mai ritentato. `MaxAttempts`ha un valore massimo di 99999999.

`BackoffRate` (facoltativo).

Il moltiplicatore con il quale l'intervallo di tentativi indicato da `IntervalSeconds` aumenta dopo ogni nuovo tentativo. Per impostazione predefinita, il valore aumenta di. `BackoffRate` `2.0`

Ad esempio, supponiamo che il tuo `IntervalSeconds` `MaxAttempts` sia 3, sia 3 e `BackoffRate` sia 2. Il primo tentativo viene eseguito tre secondi dopo il verificarsi dell'errore. Il secondo tentativo ha luogo sei secondi dopo il primo tentativo. Mentre il terzo tentativo ha luogo 12 secondi dopo il secondo tentativo.

`MaxDelaySeconds` (facoltativo).

Un numero intero positivo che imposta il valore massimo, in secondi, fino al quale un intervallo di tentativi può aumentare. Questo campo è utile da usare con il campo. `BackoffRate` Il valore specificato in questo campo limita i tempi di attesa esponenziali risultanti dal moltiplicatore della frequenza di backoff applicato a ogni tentativo consecutivo. È necessario specificare un valore maggiore di 0 e minore di 31622401 per. `MaxDelaySeconds`

Se non si specifica questo valore, Step Functions non limita i tempi di attesa tra un tentativo e l'altro.

`JitterStrategy` (facoltativo).

Una stringa che determina se includere o meno il jitter nei tempi di attesa tra tentativi consecutivi. Il jitter riduce i tentativi simultanei di nuovi tentativi distribuendoli su un intervallo di ritardo randomizzato. Questa stringa accetta or come valori. `FULL` `NONE` Il valore predefinito è `NONE`.

Ad esempio, supponiamo di aver impostato `MaxAttempts` come 3, `IntervalSeconds` come 2 e `BackoffRate` come 2. Il primo tentativo viene eseguito due secondi dopo il verificarsi dell'errore. Il secondo tentativo ha luogo quattro secondi dopo il primo tentativo e il terzo tentativo ha luogo otto secondi dopo il secondo tentativo. Se si imposta su `JitterStrategyFULL`, il primo intervallo di tentativi viene randomizzato tra 0 e 2 secondi, il secondo intervallo di tentativi viene randomizzato tra 0 e 4 secondi e il terzo intervallo tra 0 e 8 secondi.

Esempi di campi Riprova

Questa sezione include i seguenti esempi di `Retry` campo.

- Retry with BackoffRate
- Retry with MaxDelaySeconds
- Retry all errors except States.Timeout
- Complex retry scenario

> **Tip**
>
> Per implementare un esempio di flusso di lavoro per la gestione degli errori nel tuoAccount AWS, consulta il modulo Error Handling di The AWS Step Functions Workshop.

Esempio 1 — Riprova con BackoffRate

L'esempio seguente di a `Retry` effettua due tentativi di nuovo tentativo, il primo dopo aver atteso tre secondi. In base a `BackoffRate` quanto specificato, Step Functions aumenta l'intervallo tra ogni tentativo fino al raggiungimento del numero massimo di tentativi. Nell'esempio seguente, il secondo tentativo inizia dopo aver atteso tre secondi dopo il primo tentativo.

```
"Retry": [ {
   "ErrorEquals": [ "States.Timeout" ],
   "IntervalSeconds": 3,
   "MaxAttempts": 2,
   "BackoffRate": 1
} ]
```

Esempio 2 — Riprova con MaxDelaySeconds

L'esempio seguente esegue tre tentativi e limita il tempo di attesa risultante `BackoffRate` a 5 secondi. Il primo tentativo viene eseguito dopo un'attesa di tre secondi. Il secondo e il terzo tentativo hanno luogo dopo aver atteso cinque secondi dopo il tentativo precedente a causa del limite massimo di attesa impostato da. `MaxDelaySeconds`

```
"Retry": [ {
    "ErrorEquals": [ "States.Timeout" ],
    "IntervalSeconds": 3,
    "MaxAttempts": 3,
    "BackoffRate":2,
    "MaxDelaySeconds": 5,
    "JitterStrategy": "FULL"
} ]
```

In caso contrario`MaxDelaySeconds`, il secondo tentativo avrebbe luogo sei secondi dopo il primo e il terzo tentativo 12 secondi dopo il secondo.

Esempio 3: riprova tutti gli errori tranne States.Timeout

Il nome riservato `States.ALL` che compare nel campo `ErrorEquals` di un retrier è un carattere jolly che corrisponde a qualsiasi nome di errore. Deve apparire da solo nell'array `ErrorEquals` e deve figurare nell'ultimo retrier dell'array `Retry`. Il nome funge `States.TaskFailed` anche da jolly e corrisponde a qualsiasi errore tranne. `States.Timeout`

Il seguente esempio di `Retry` campo riprova qualsiasi errore tranne. `States.Timeout`

```
"Retry": [ {
   "ErrorEquals": [ "States.Timeout" ],
   "MaxAttempts": 0
}, {
   "ErrorEquals": [ "States.ALL" ]
} ]
```

Esempio 4 — Scenario di tentativi complessi

I parametri di un retrier si applicano a tutte le visite al retrier nel contesto di un'esecuzione a stato singolo.

Considera lo stato `Task` seguente.

```
"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"
   } ]
}
```

Questa attività ha esito negativo quattro volte di seguito, e vengono restituiti i seguenti nomi di errore:`ErrorA`,`ErrorB`, `ErrorC` e. `ErrorB` Il risultato è il seguente:

- I primi due errori corrispondono al primo retrier e causano attese di uno e due secondi.
- Il terzo errore corrisponde al secondo retrier e causa un'attesa di cinque secondi.
- Il quarto errore corrisponde anche al primo retrier. Tuttavia, ha già raggiunto il numero massimo di due tentativi (`MaxAttempts`) per quel particolare errore. Pertanto, quel retrier ha esito negativo e l'esecuzione reindirizza il flusso di lavoro `Z` allo stato tramite il campo. `Catch`

Stati di fallback

`Task`, `Map` e `Parallel` gli stati possono avere ciascuno un campo denominato`Catch`. Il valore di questo campo deve essere un array di oggetti conosciuti come catchers.

Un catcher contiene i seguenti campi.

`ErrorEquals` (Obbligatorio)

Un array non vuoto di stringhe che corrispondono a nomi di errore, specificati in modo identico al campo del retrier con il medesimo nome.

`Next` (Obbligatorio)

Una stringa che deve corrispondere esattamente a uno dei nomi di stato della macchina a stati.

`ResultPath` (facoltativo).

Un percorso che determina l'input che il catcher invia allo stato specificato nel `Next` campo.

Quando uno stato segnala un errore e non è presente alcun `Retry` campo o se i nuovi tentativi non riescono a risolvere l'errore, Step Functions analizza i catcher nell'ordine elencato nell'array. Quando il nome dell'errore compare nel valore del campo `ErrorEquals` di un catcher, la macchina a stati passa allo stato denominato nel campo `Next`.

Il nome riservato `States.ALL` che compare nel campo `ErrorEquals` di un catcher è un carattere jolly che corrisponde a qualsiasi nome di errore. Deve apparire da solo nell'array `ErrorEquals` e deve figurare nell'ultimo catcher dell'array `Catch`. Il nome funge `States.TaskFailed` anche da jolly e corrisponde a qualsiasi errore tranne. `States.Timeout`

Il seguente esempio di un `Catch` campo passa allo stato denominato `RecoveryState` quando una funzione Lambda genera un'eccezione Java non gestita. In caso contrario, il campo passa allo stato `EndState`.

```
"Catch": [ {
   "ErrorEquals": [ "java.lang.Exception" ],
   "ResultPath": "$.error-info",
   "Next": "RecoveryState"
}, {
   "ErrorEquals": [ "States.ALL" ],
   "Next": "EndState"
} ]
```

> **Note**
>
> Ogni catcher può specificare più errori da gestire.

Output dell'errore

Quando Step Functions passa allo stato specificato in un catch name, l'oggetto di solito contiene il campo`Cause`. Il valore di questo campo è una descrizione leggibile dell'errore. L'oggetto è noto come output dell'errore.

Nel presente esempio il primo catcher contiene un campo `ResultPath`. Questo funziona in modo analogo a un campo `ResultPath` nel livello superiore di uno stato e genera due possibilità:

- Prende i risultati dell'esecuzione di quello stato e sovrascrive tutto o parte dell'input dello stato.
- Prende i risultati e li aggiunge all'input. Nel caso di un errore gestito da un catcher, il risultato dell'esecuzione dello stato è l'output dell'errore.

Pertanto, per il primo catcher dell'esempio, il catcher aggiunge l'output di errore all'input come campo denominato `error-info` se nell'input non è già presente un campo con questo nome. Quindi, il catcher invia l'intero input a. `RecoveryState` Per il secondo ricevitore, l'output di errore sovrascrive l'input e il catcher invia solo l'output di errore a. `EndState`

> **Note**
>
> Se non specifichi il campo `ResultPath`, l'impostazione predefinita è $, che seleziona e sovrascrive l'intero input.

Quando uno stato ha entrambi `Catch` i campi `Retry` e, Step Functions utilizza prima tutti i retrier appropriati. Se la politica di riprova non riesce a risolvere l'errore, Step Functions applica la transizione matching catcher.

Causa payload e integrazioni di servizi

Un catcher restituisce un payload di stringa come output. Quando lavori con integrazioni di servizi come Amazon Athena AWS CodeBuild o, potresti voler convertire `Cause` la stringa in JSON. Il seguente esempio di `Pass` stato con funzioni intrinseche mostra come convertire una stringa in JSON. `Cause`

```
"Handle escaped JSON with JSONtoString": {
  "Type": "Pass",
  "Parameters": {
    "Cause.$": "States.StringToJson($.Cause)"
  },
  "Next": "Pass State with Pass Processing"
},
```

Esempi di macchine a stati che utilizzano Retry e Catch

Le macchine a stati definite negli esempi seguenti presuppongono l'esistenza di due funzioni Lambda: una che fallisce sempre e una che attende abbastanza a lungo da consentire il verificarsi di un timeout definito nella macchina a stati.

Questa è una definizione di una funzione Lambda di Node.js che fallisce sempre e restituisce il messaggio. `error` Negli esempi di macchine a stati che seguono, questa funzione Lambda è denominata. `FailFunction` Per informazioni sulla creazione di una funzione Lambda, vedere Fase 1: Creazione di una funzione Lambda la sezione.

```
exports.handler = (event, context, callback) => {
    callback("error");
};
```

Questa è una definizione di una funzione Lambda di Node.js che rimane inattiva per 10 secondi. Negli esempi di macchine a stati che seguono, questa funzione Lambda è denominata. `sleep10`

> **Note**
>
> Quando crei questa funzione Lambda nella console Lambda, ricordati di modificare il valore di Timeout nella sezione Impostazioni avanzate da 3 secondi (impostazione predefinita) a 11 secondi.

```
exports.handler = (event, context, callback) => {
    setTimeout(function(){
    }, 11000);
};
```

Gestione di un errore utilizzando Retry

Questa macchina a stati utilizza un campo `Retry` per ritentare una funzione che ha esito negativo e genera il nome di errore `HandledError`. Riprova questa funzione due volte con un backoff esponenziale tra un tentativo e l'altro.

```
{
   "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
      }
   }
}
```

Questa variante utilizza il codice di errore predefinito`States.TaskFailed`, che corrisponde a qualsiasi errore generato da una funzione Lambda.

```
{
   "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**
>
> Come best practice, le attività che fanno riferimento a una funzione Lambda dovrebbero gestire le eccezioni del servizio Lambda. Per ulteriori informazioni, consulta Gestire le eccezioni del servizio Lambda.

Gestione di un errore utilizzando Catch

In questo esempio viene utilizzato un campo `Catch`. Quando una funzione Lambda emette un errore, rileva l'errore e la macchina a stati passa allo stato. `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
      }
   }
}
```

Questa variante utilizza il codice di errore predefinito`States.TaskFailed`, che corrisponde a qualsiasi errore generato da una funzione Lambda.

```
{
   "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
      }
   }
}
```

Gestione di un timeout utilizzando Retry

Questa macchina a stati utilizza un `Retry` campo per riprovare uno `Task` stato scaduto, in base al valore di timeout specificato in. `TimeoutSeconds` Step Functions ritenta l'invocazione della funzione Lambda in `Task` questo stato due volte, con un backoff esponenziale tra un tentativo e l'altro.

```
{
   "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
      }
   }
}
```

Gestione di un timeout utilizzando Catch

In questo esempio viene utilizzato un campo `Catch`. Quando si verifica un timeout, la macchina a stati passa allo stato `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**
>
> Puoi conservare l'input di stato e l'errore utilizzando `ResultPath`. Consulta ResultPath Da utilizzare per includere sia l'errore che l'input in un `Catch`.

InvocareAWS Step Functionsda altri servizi

È possibile configurare diversi altri servizi per richiamare macchine a stati. Basato su quello della macchina a statitipo di flusso di lavoro, è possibile richiamare macchine a stati in modo asincrono o sincrono. Per richiamare le macchine a stati in modo sincrono, utilizzare`StartSyncExecution`Chiamata API o integrazione di Amazon API Gateway con Express Workflows. Con l'invocazione asincrona, Step Functions sospende l'esecuzione del flusso di lavoro fino alla restituzione di un task token. Tuttavia, l'attesa di un task token rende il flusso di lavoro sincrono.

I servizi che è possibile configurare per richiamare Step Functions includono:

- AWS Lambda, utilizzando il`StartExecution`chiamata.

- Gateway Amazon API
- AmazonEventBridge
- AWS CodePipeline
- AWS IoTMotore di regole
- AWS Step Functions

Le chiamate di Step Functions sono regolate da`StartExecution`quota. Per ulteriori informazioni, consultare:

- Quote

Leggi Coerency in Step Functions

Gli aggiornamenti delle macchine a stati in AWS Step Functions hanno consistenza finale. Tutte le `StartExecution` chiamate entro pochi secondi utilizzeranno la definizione aggiornata e `roleArn` (Amazon Resource Name per il ruolo IAM). Le esecuzioni avviate immediatamente dopo la chiamata di `UpdateStateMachine` potrebbero utilizzare la definizione della macchina a stati e il valore `roleArn` precedenti.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- `UpdateStateMachine` nel documento di riferimento delle API AWS Step Functions
- Aggiorna un flusso di lavoro in. Nozioni di base su AWS Step Functions

Inserimento di tag in Step Functions

AWS Step Functions supporta il tagging di macchine a stati (sia standard che Express) e attività. Questo può aiutarti a monitorare e gestire i costi associati alle tue risorse e a fornire una maggiore sicurezza nelle tue policy AWS Identity and Access Management (IAM). La codifica delle risorse di Step Functions consente di gestirle daAWS Resource Groups. Per ulteriori informazioni sui gruppi di risorse, consulta la Guida per AWS Resource Groups l'utente.

Per l'autorizzazione basata su tag, le risorse di esecuzione della macchina a stati, come mostrato nell'esempio seguente, ereditano i tag associati a una macchina a stati.

```
arn:<partition>:states:<Region>:<account-id>:execution:<StateMachineName>:<ExecutionId>
```

Quando chiami DescribeExecutiono altre API in cui specifichi la risorsa di esecuzione ARN, Step Functions utilizza i tag associati alla macchina a stati per accettare o rifiutare la richiesta durante l'esecuzione dell'autorizzazione basata su tag. Ciò consente di consentire o negare l'accesso alle esecuzioni delle macchine a stati a livello di macchina a stati.

Per esaminare le limitazioni correlate la tagging delle risorse, consultare Restrizioni relative all'etichettatura.

Argomenti

- Tagging per l'allocazione dei costi
- Tagging per la sicurezza
- Visualizzazione e gestione dei tag nella console Step Functions
- Gestisci i tag con le azioni API di Step Functions

Tagging per l'allocazione dei costi

Per organizzare e identificare le risorse di Step Functions per l'allocazione dei costi, è possibile aggiungere tag di metadati che identificano lo scopo di una macchina statale o di un'attività. Questo è particolarmente utile quando si dispone di numerose risorse. Per organizzare le fatture AWS al fine di riflettere la struttura dei costi, puoi utilizzare i tag di allocazione dei costi. Per eseguire questa operazione, registrati per far sì che la fattura del tuo account AWS includa le chiavi e i valori di tag. Per ulteriori informazioni, consulta Impostazione di un report di allocazione dei costi mensili nella Guida per l'utente di AWS Billing.

Ad esempio, è possibile aggiungere tag che rappresentano il centro di costo e lo scopo delle risorse di Step Functions, come segue.

Risorsa	Key (Chiave)	Value (Valore)
`StateMachine1`	`Cost Center`	`34567`
	`Application`	`Image processing`
`StateMachine2`	`Cost Center`	`34567`
	`Application`	`Rekognition processing`

Risorsa	Key (Chiave)	Value (Valore)
`Activity1`	`Cost Center`	`12345`
	`Application`	`Legacy database`

Questo schema di tagging consente di raggruppare due macchine a stati che eseguono task correlati nello stesso centro di costo, mentre si esegue il tagging di un'attività non pertinente con un tag di allocazione dei costi differente.

Tagging per la sicurezza

IAM supporta il controllo dell'accesso alle risorse in base ai tag. Per controllare l'accesso in base ai tag, fornisci informazioni sui tag delle risorse nell'elemento condizionale di una policy IAM.

Ad esempio, è possibile limitare l'accesso a tutte le risorse di Step Functions che includono un tag con la chiave `environment` e il valore`production`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "states:TagResource",
                "states:DeleteActivity",
                "states:DeleteStateMachine",
                "states:StopExecution"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/environment": "production"}
            }
        }
    ]
}
```

Per ulteriori informazioni, consulta Controllo dell'accesso tramite tag nella Guida per l'utente di IAM.

Visualizzazione e gestione dei tag nella console Step Functions

Step Functions consente di visualizzare e gestire i tag per le macchine a stato nella console Step Functions. Dalla pagina Details (Dettagli) di una macchina a stati, seleziona Tags (Tag). Qui è possibile visualizzare i tag esistenti associati alla macchina a stati.

> **Note**
>
> Per gestire i tag per le attività, consultare Gestisci i tag con le azioni API di Step Functions.

Per aggiungere o eliminare i tag che sono associati alla macchina di stato, seleziona il pulsante Manage Tags (Gestisci tag).

1. Vai alla pagina dei dettagli di una macchina a stati.
2. Seleziona Tags (Tag) accanto a Executions (Esecuzioni) eDefinition (Definizione).
3. Scegliere Manage tags (Gestisci tag).
   - Per modificare i tag esistenti, modifica Key (Chiave) e Value (Valore).
   - Per rimuovere i tag esistenti, scegli Remove tag (Rimuovi tag).
   - Per aggiungere un nuovo tag, scegli Add tag (Aggiungi tag) e immetti Key (Chiave) e Value (Valore).
4. Seleziona Salva.

Gestisci i tag con le azioni API di Step Functions

Per gestire i tag utilizzando l'API Step Functions, utilizza le seguenti azioni API:

- `ListTagsForResource`
- `TagResource`
- `UntagResource`

AWS Step Functions Workflow Studio

Workflow Studio for AWS Step Functions è un designer di flussi di lavoro visivi a basso codice che consente di creare flussi di lavoro senza server orchestrando i servizi. AWS Utilizzando la sua drag-and-drop funzionalità o l'editor di codice integrato, è possibile creare e modificare flussi di lavoro, controllare come l'input e l'output vengono filtrati o trasformati per ogni stato e configurare la gestione degli errori. Mentre trascini gli stati per creare il tuo flusso di lavoro, Workflow Studio convalida il tuo lavoro e genera automaticamente il codice. Puoi rivedere il codice generato o aggiornare la definizione della macchina a stati all'interno dell'editor di codice. Al termine, puoi salvare il flusso di lavoro, eseguirlo, quindi esaminare i risultati nella console Step Functions. Puoi aggiungere e modificare visivamente i flussi di lavoro per orchestrare i molteplici servizi dell'applicazione.

Per utilizzare Step Functions Workflow Studio, avrai bisogno di un Account AWS, e delle credenziali che forniscano le autorizzazioni corrette per tutte le risorse che desideri utilizzare. Per ulteriori informazioni, consulta Prerequisiti per iniziare con AWS Step Functions.

> **Note**
>
> Workflow Studio non supporta Internet Explorer 11. Se utilizzi Internet Explorer 11 e riscontri problemi con Workflow Studio, prova a utilizzare un browser diverso.

Puoi accedere a Workflow Studio dalla console Step Functions, quando crei o modifichi un flusso di lavoro in Step Functions. Puoi accedere a Workflow Studio anche daStrumento AWS per la creazione di applicazioni. Workflow Studio in Application Composer fornisce un ambiente visivo IaC che semplifica l'integrazione dei flussi di lavoro nelle applicazioni serverless create utilizzando strumenti IaC, come i modelli. AWS CloudFormation Utilizzando Workflow Studio inApplication Composer, è possibile creare flussi di lavoro utilizzando modelli. AWS CloudFormation All'internoApplication Composer, è possibile aggiungere un nuovo flusso di lavoro, modificare un flusso di lavoro esistente e collegare le singole fasi del flusso di lavoro ad altre risorse dell'applicazione. Application Composercrea e aggiorna automaticamente le CloudFormation risorse e le configurazioni necessarie. Questo ti aiuta a creare e gestire tutte le risorse utilizzate nei flussi di lavoro in un unico posto. Ciò consente inoltre di accelerare il percorso dalla prototipazione del flusso di lavoro all'implementazione in produzione.

Quando utilizzi Workflow Studio inApplication Composer, puoi anche connetterti direttamente al tuo progetto locale. Questo ti aiuta a lavorare nel tuo ambiente di sviluppo integrato (IDE) insieme

al visual canvas. Per ulteriori informazioni, consulta Utilizzo di Workflow Studio in Application Composer.

Argomenti

- Panoramica dell'interfaccia
- Utilizzo di Workflow Studio
- Configura ingressi e uscite per i tuoi stati
- Ruoli di esecuzione in Workflow Studio
- Gestione degli errori
- Tutorial: Impara a usare AWS Step Functions Workflow Studio

Panoramica dell'interfaccia

Workflow Studio for AWS Step Functions è un designer di flussi di lavoro visivo a basso codice che consente di creare flussi di lavoro senza server orchestrando. Servizi AWS Con la sua funzione drag and drop, Workflow Studio semplifica la creazione, la modifica e la visualizzazione dei prototipi del flusso di lavoro. Workflow Studio offre anche un editor di codice integrato per scrivere e modificare le definizioni del flusso di lavoro utilizzando Amazon States Language (ASL) all'interno della console Step Functions.

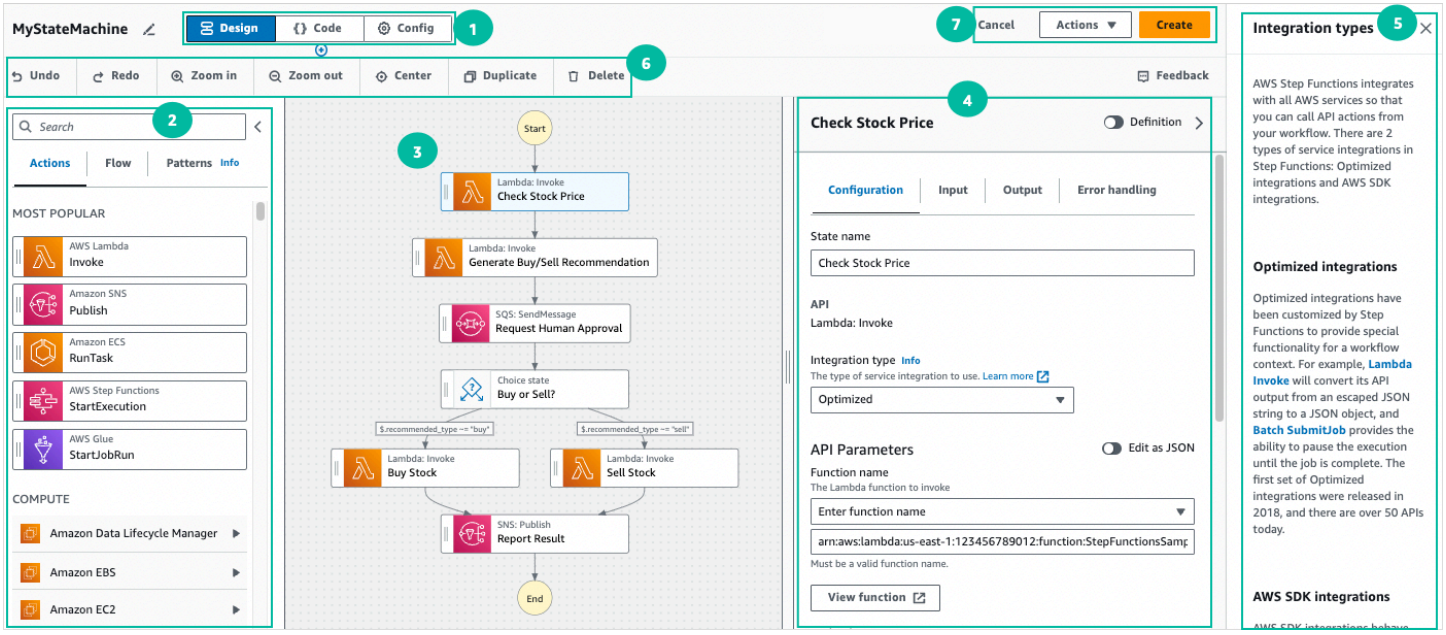
Per aiutarti a creare e visualizzare i flussi di lavoro, modificarne le definizioni e gestirne la configurazione, Workflow Studio offre tre modalità: Design, Code e Config. Le sezioni seguenti descrivono queste modalità in dettaglio.

In questo argomento

- Modalità di progettazione
- Modalità codice
- modalità Config
- Tasti di scelta rapida

Modalità di progettazione

La modalità Progettazione di Workflow Studio fornisce un'interfaccia grafica per visualizzare i flussi di lavoro durante la creazione dei prototipi. L'immagine seguente mostra i diversi componenti disponibili nella modalità Progettazione.

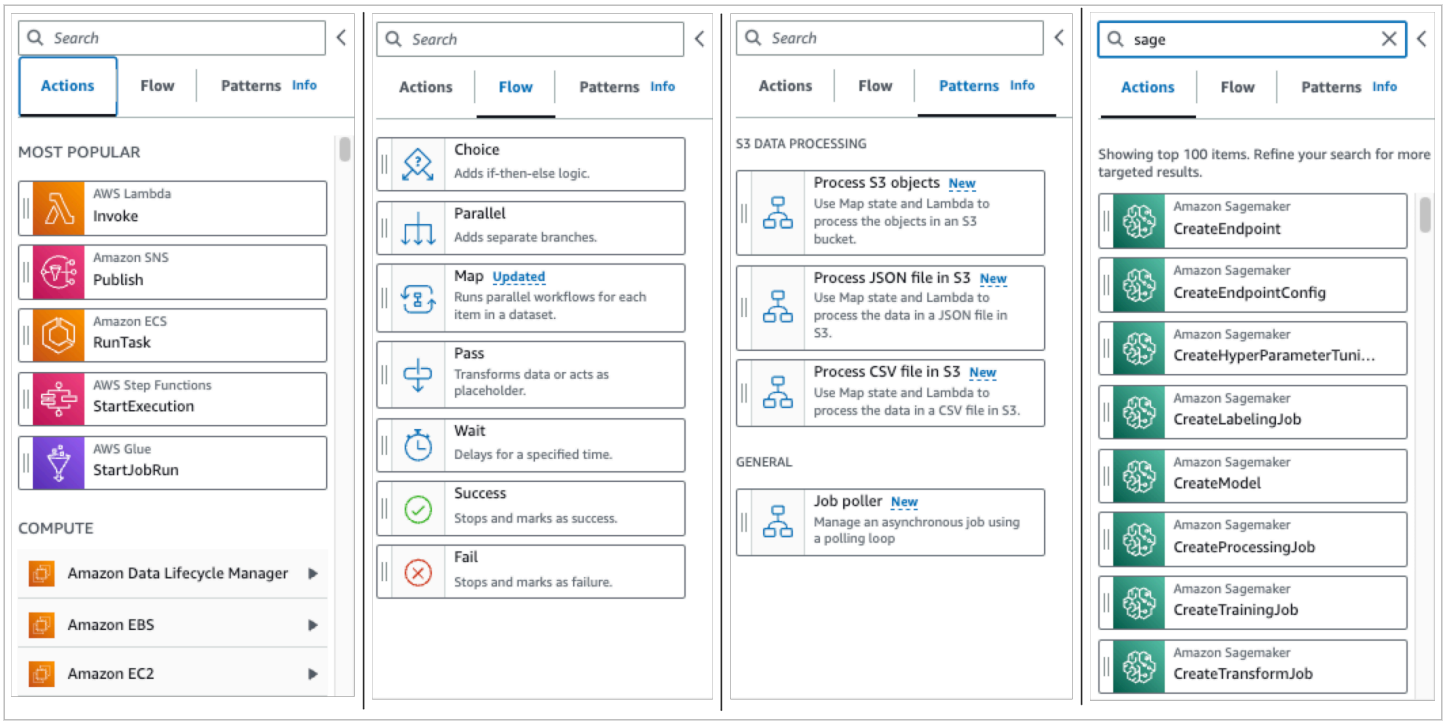


1. Pulsanti di modalità: passa tra le modalità Design, Code e Config di Workflow Studio utilizzando i pulsanti di modalità. Non puoi cambiare modalità se il codice JSON nella definizione ASL del flusso di lavoro non è valido.
2. Stati del browserContiene le seguenti tre schede:
   - La scheda Azioni fornisce un elenco di AWS API che puoi trascinare nel grafico del flusso di lavoro nell'area di disegno. Ogni azione rappresenta uno Attività stato.
   - La scheda Flusso fornisce un elenco di stati del flusso che puoi trascinare nel grafico del flusso di lavoro nell'area di disegno.
   - La scheda Patterns fornisce diversi ready-to-use blocchi costitutivi riutilizzabili che puoi utilizzare per una varietà di casi d'uso. Ad esempio, puoi utilizzare questi modelli per elaborare in modo iterativo i dati in un bucket Amazon S3.
3. CanvasÈ qui che puoi trascinare gli stati nel grafico del flusso di lavoro, modificare l'ordine degli stati e selezionare gli stati da configurare o visualizzare.
4. Il ✓Inspector pannello consente di visualizzare e modificare le proprietà di qualsiasi stato selezionato nell'area di disegno. Attiva l'interruttore Definition per visualizzare il codice Amazon States Language per il tuo flusso di lavoro, con lo stato attualmente selezionato evidenziato.
5. I link informativi aprono un pannello con informazioni contestuali quando hai bisogno di aiuto. Questi pannelli includono anche collegamenti ad argomenti correlati nella documentazione di Step Functions.
6. Barra degli strumenti di progettazione: contiene una serie di pulsanti per eseguire azioni comuni, come annullare, eliminare e ingrandire.

7. Pulsanti di utilità: un set di pulsanti per eseguire attività, come il salvataggio dei flussi di lavoro o l'esportazione delle relative definizioni ASL in un file JSON o YAML.

Stati del browser

Il browser States consente di selezionare gli stati da trascinare nel grafico del flusso di lavoro. La scheda Azioni fornisce un elenco di AWS API e la scheda Flusso fornisce un elenco di stati del flusso. Mentre la scheda Patterns fornisce diversi ready-to-use blocchi predefiniti riutilizzabili che è possibile utilizzare per una varietà di casi d'uso. Puoi cercare tutti gli stati nello States Browser utilizzando la casella di ricerca in alto.



Esistono sette stati di flusso che è possibile utilizzare per indirizzare e controllare il flusso di lavoro. Tutti prendono input dallo stato precedente e molti consentono di filtrare l'input dallo stato precedente e l'output dallo stato successivo. Gli stati del flusso sono:

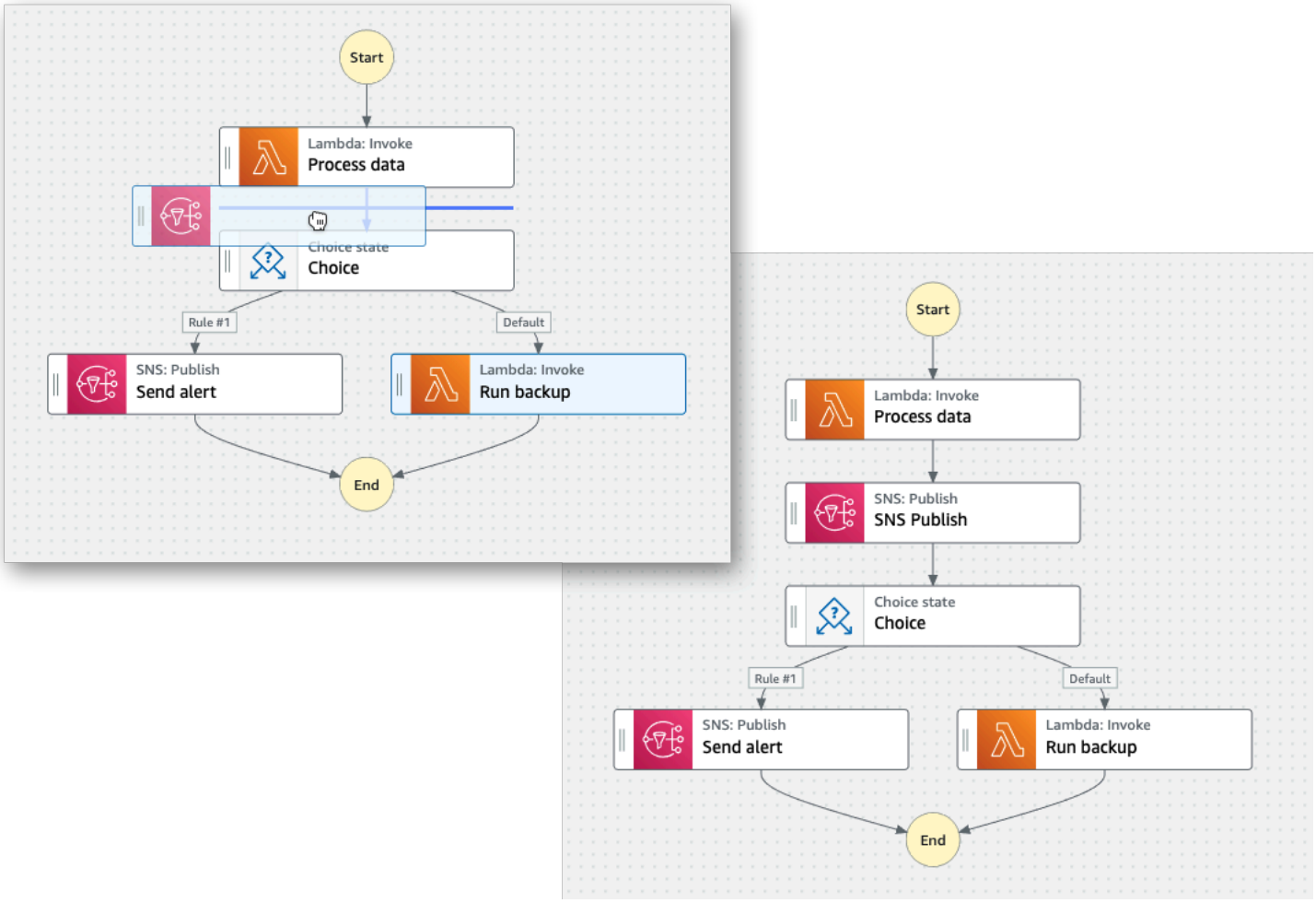
- Choice: aggiungi una scelta tra i rami di esecuzione al tuo flusso di lavoro. Nella scheda Configurazione di Inspector, puoi configurare le regole per determinare a quale stato passerà il flusso di lavoro.
- Parallel: aggiungi rami di esecuzione paralleli al tuo flusso di lavoro.

- Eseguire la mappatura: itera dinamicamente i passaggi per ogni elemento di un array di input. A differenza di uno stato di `Parallel` flusso, uno `Map` stato eseguirà gli stessi passaggi per più voci di un array nell'input di stato.
- Pass: Consente di passare l'input all'output. (Facoltativo) È possibile aggiungere dati fissi nell'output.
- Attendi: Metti in pausa il flusso di lavoro per un determinato periodo di tempo o fino a un'ora o una data specificate.
- Succeed: interrompe il flusso di lavoro con successo.
- Fail: interrompe il flusso di lavoro in caso di errore.

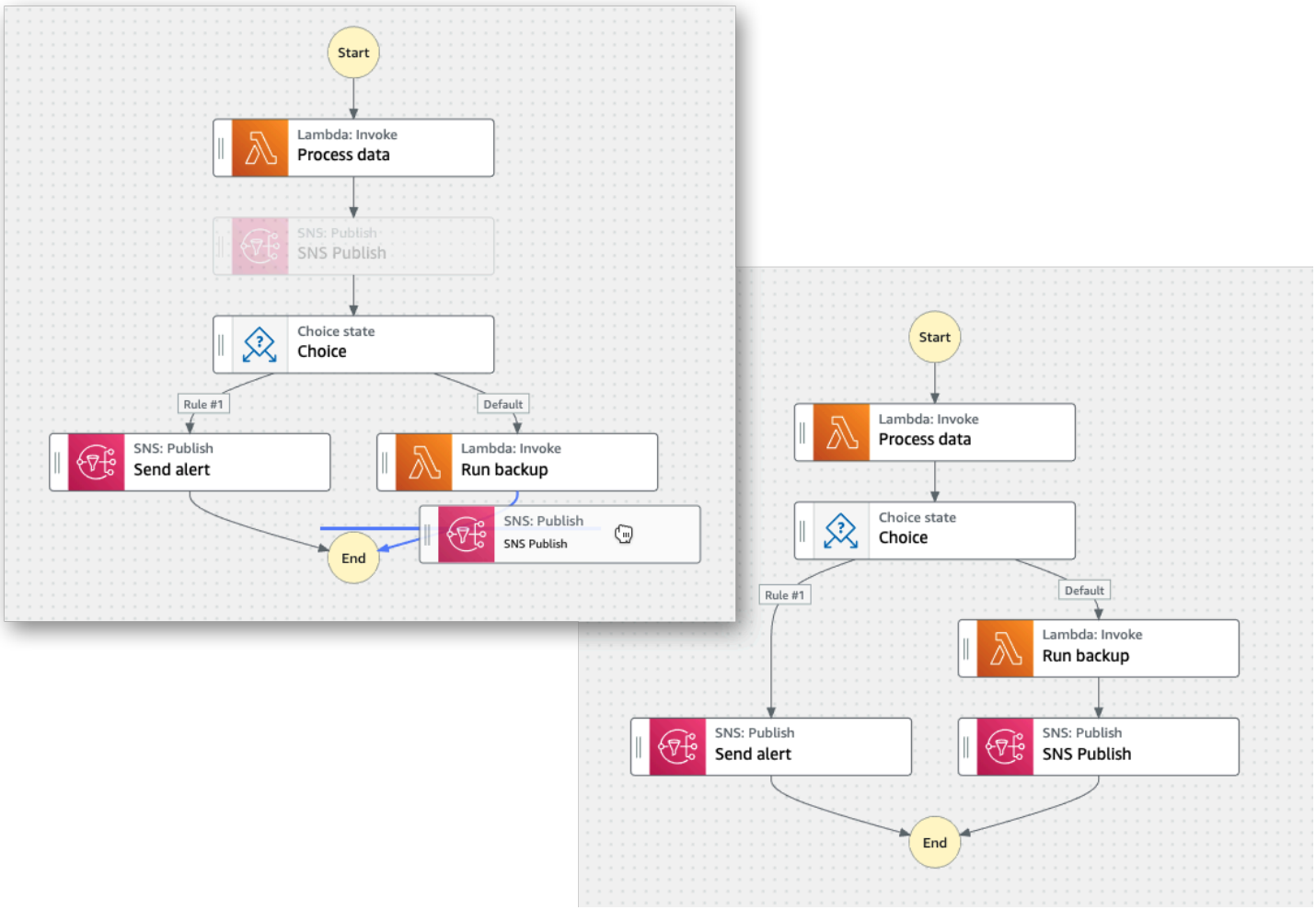
Canvas

Dopo aver scelto uno stato da aggiungere al flusso di lavoro, trascinalo nell'area di disegno e inseriscilo nel grafico del flusso di lavoro. Puoi anche trascinare gli stati per spostarli in punti diversi del flusso di lavoro. Se il flusso di lavoro è complesso, potresti non essere in grado di visualizzarlo tutto nel pannello canvas. Usa i controlli nella parte superiore dell'area di disegno per ingrandire o ridurre una selezione. Per visualizzare diverse parti di un grafico del flusso di lavoro, puoi trascinare il grafico del flusso di lavoro nell'area di disegno.

Trascina lo stato del flusso di lavoro dalla scheda Azioni o Flusso e inseriscilo nel tuo flusso di lavoro. Una riga mostra dove verrà posizionato nel flusso di lavoro. Il nuovo stato del flusso di lavoro è stato aggiunto al flusso di lavoro e il relativo codice viene generato automaticamente.

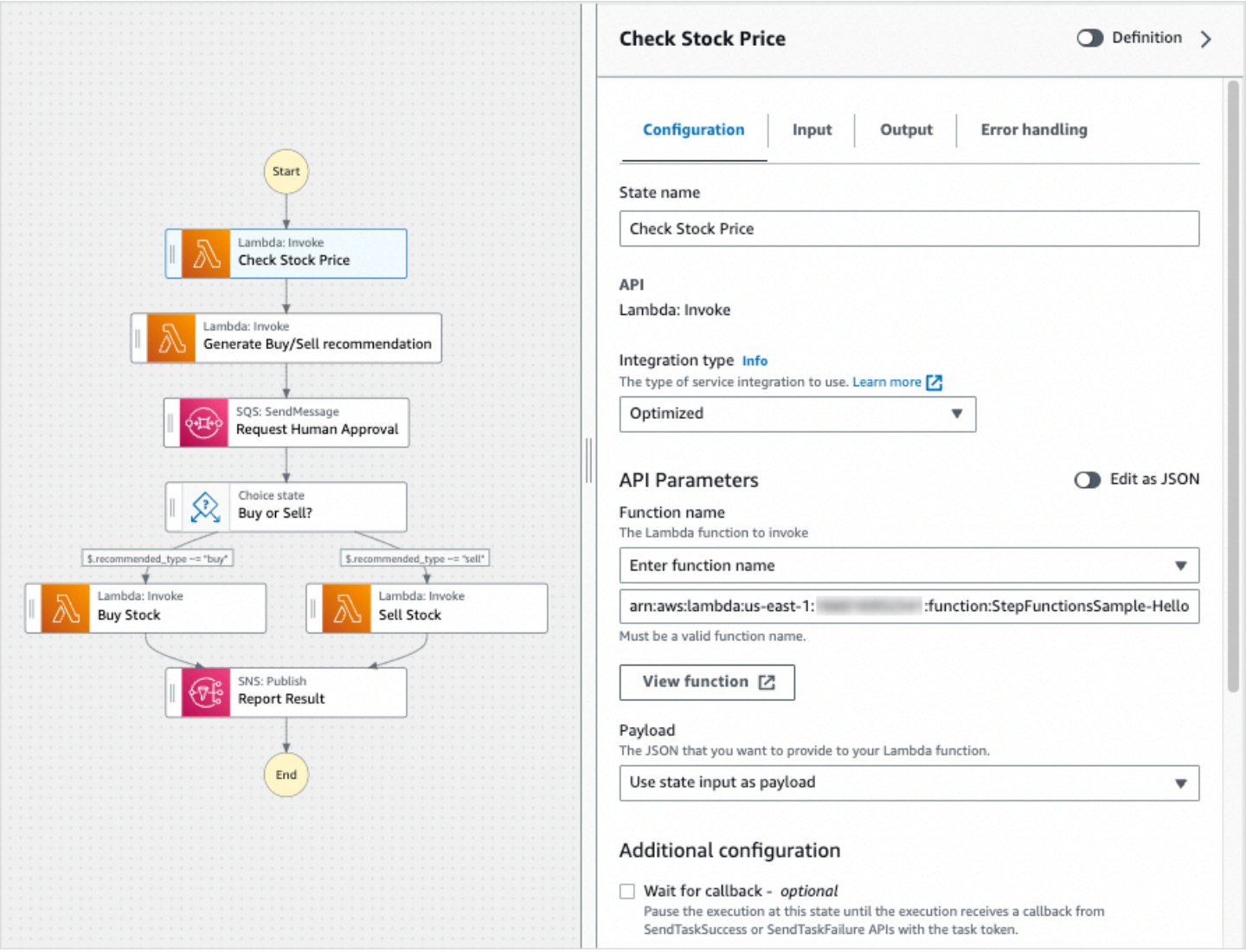


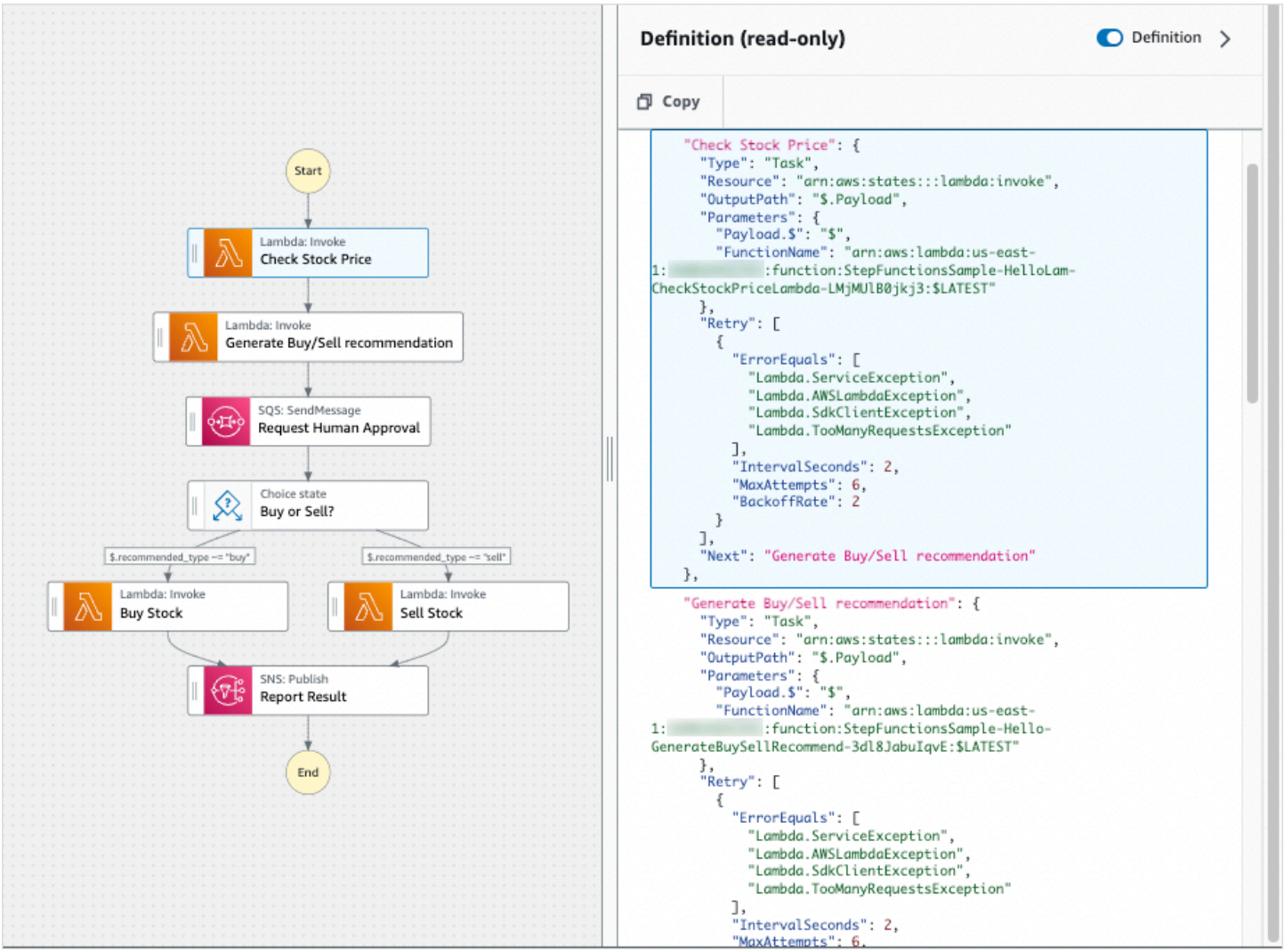
Per modificare l'ordine di uno stato, puoi trascinarlo in una posizione diversa del flusso di lavoro.



✓Inspector

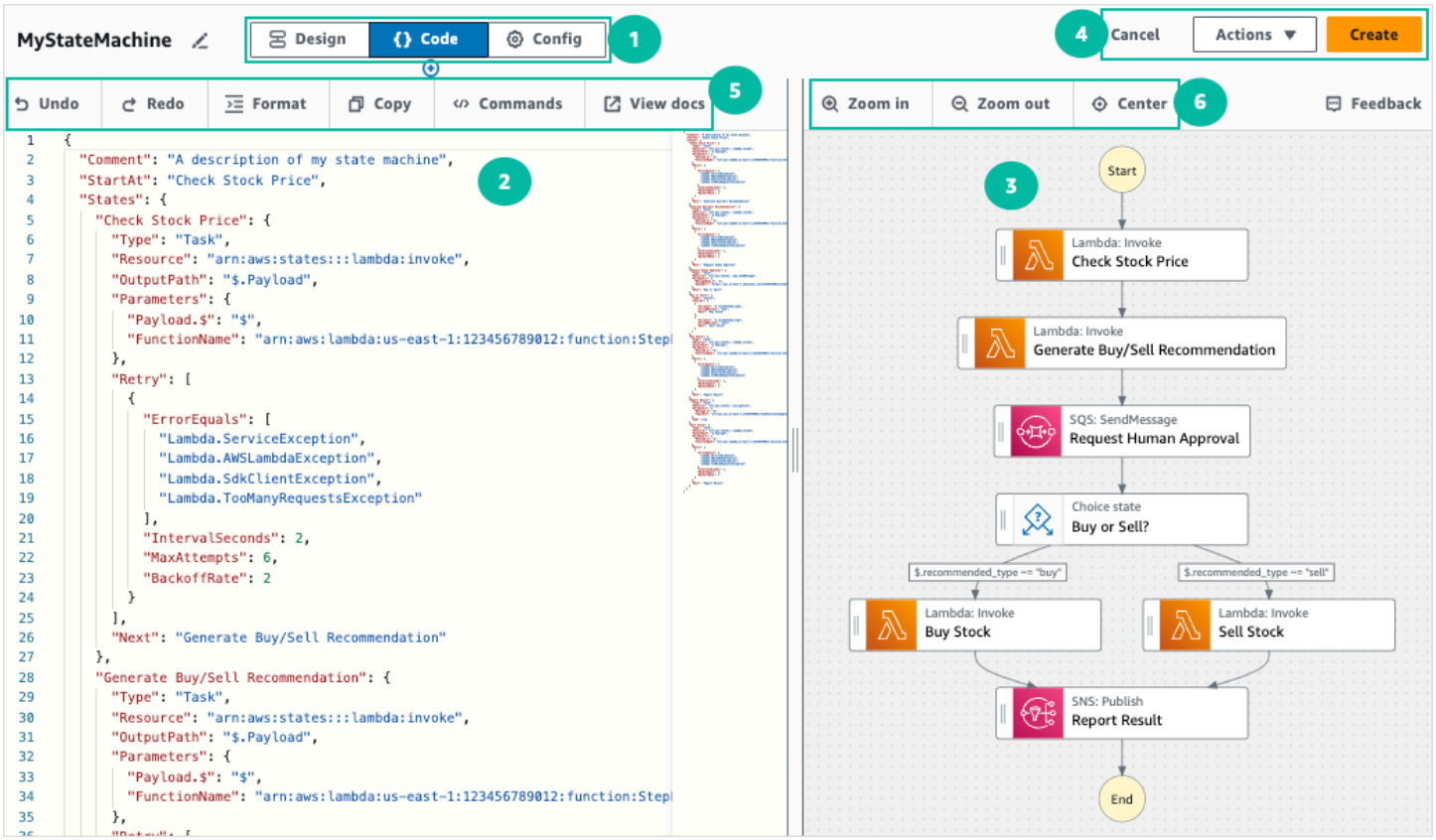
Puoi configurare tutti gli stati che aggiungi al tuo flusso di lavoro. Scegliete lo stato che desiderate configurare e vedrete le relative opzioni di configurazione nel pannello Inspector. Per visualizzare la definizione ASL generata automaticamente per il codice del flusso di lavoro, attiva l'interruttore Definizione. La definizione ASL associata allo stato selezionato apparirà evidenziata.


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.



Modalità codice

La modalità Codice di Workflow Studio fornisce un editor di codice integrato per visualizzare, scrivere e modificare la definizione Amazon States Language (ASL) dei flussi di lavoro all'interno della console Step Functions. L'immagine seguente mostra i diversi componenti disponibili nella modalità Codice.



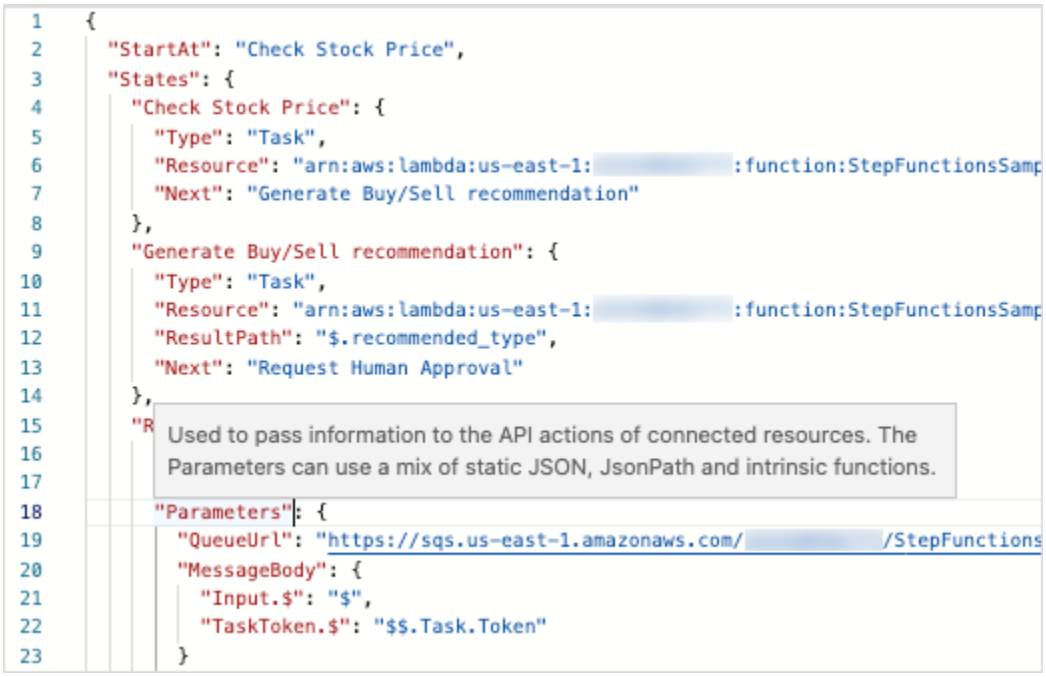
1. Pulsanti di modalità: passa tra le modalità Design, Code e Config di Workflow Studio utilizzando i pulsanti di modalità. Non puoi cambiare modalità se il codice JSON nella definizione ASL del flusso di lavoro non è valido.
2. Editor del codiceÈ dove scrivi e modifichi la definizione ASL dei tuoi flussi di lavoro all'interno di Workflow Studio. L'editor di codice fornisce anche funzionalità come l'evidenziazione della sintassi e il completamento automatico.
3. Riquadro di visualizzazione del grafico— Mostra una visualizzazione grafica in tempo reale del flusso di lavoro.
4. Pulsanti di utilità: un set di pulsanti per eseguire attività, come il salvataggio dei flussi di lavoro o l'esportazione delle relative definizioni ASL in un file JSON o YAML.
5. Barra degli strumenti del codice: contiene una serie di pulsanti per eseguire azioni comuni, come annullare un'azione o formattare il codice.
6. Barra degli strumenti del grafico: contiene una serie di pulsanti per eseguire azioni comuni, come ingrandire e rimpicciolire il grafico del flusso di lavoro.

Editor del codice

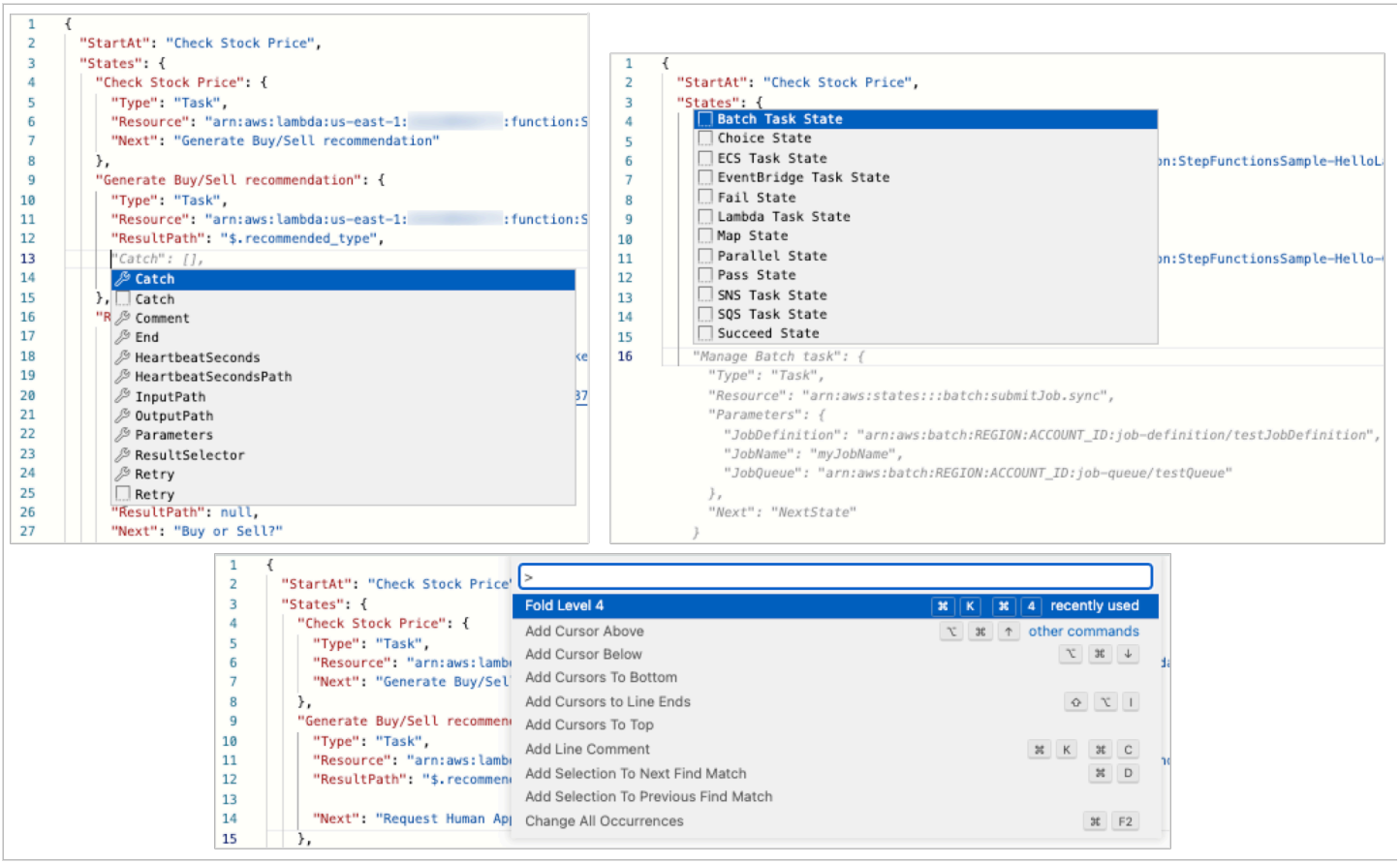
L'editor di codice offre un'esperienza simile a un IDE per scrivere e modificare le definizioni del flusso di lavoro utilizzando JSON all'interno di Workflow Studio. L'editor di codice include diverse funzionalità, come l'evidenziazione della sintassi, i suggerimenti di completamento automatico, la convalida delle definizioni ASL e la visualizzazione della guida sensibile al contesto. Man mano che aggiorni la definizione del flusso di lavoro, Riquadro di visualizzazione del grafico viene visualizzato un grafico in tempo reale del flusso di lavoro. Puoi anche vedere il grafico aggiornato del flusso di lavoro nelModalità di progettazione.

Se si seleziona uno stato nel riquadro di visualizzazione del grafico Modalità di progettazione o nel riquadro di visualizzazione del grafico, la definizione ASL di tale stato appare evidenziata nell'editor di codice. La definizione ASL del flusso di lavoro viene aggiornata automaticamente se riordini, elimini o aggiungi uno stato nella modalità Progettazione o nel riquadro di visualizzazione del grafico.

Posiziona il cursore su qualsiasi campo della definizione del flusso di lavoro per visualizzarne la guida sensibile al contesto come descrizione.



I suggerimenti di completamento automatico mostrano frammenti di codice per i campi o gli stati che puoi includere nei tuoi flussi di lavoro. Per visualizzare un elenco di campi che puoi includere in uno stato specifico, premi. **`Ctrl+Space`** Per generare un frammento di codice per un nuovo stato nel flusso di lavoro, premi **`Ctrl+Space`** dopo la definizione dello stato corrente. Puoi anche premere **`F1`** per visualizzare un elenco di comandi disponibili.



Riquadro di visualizzazione del grafico

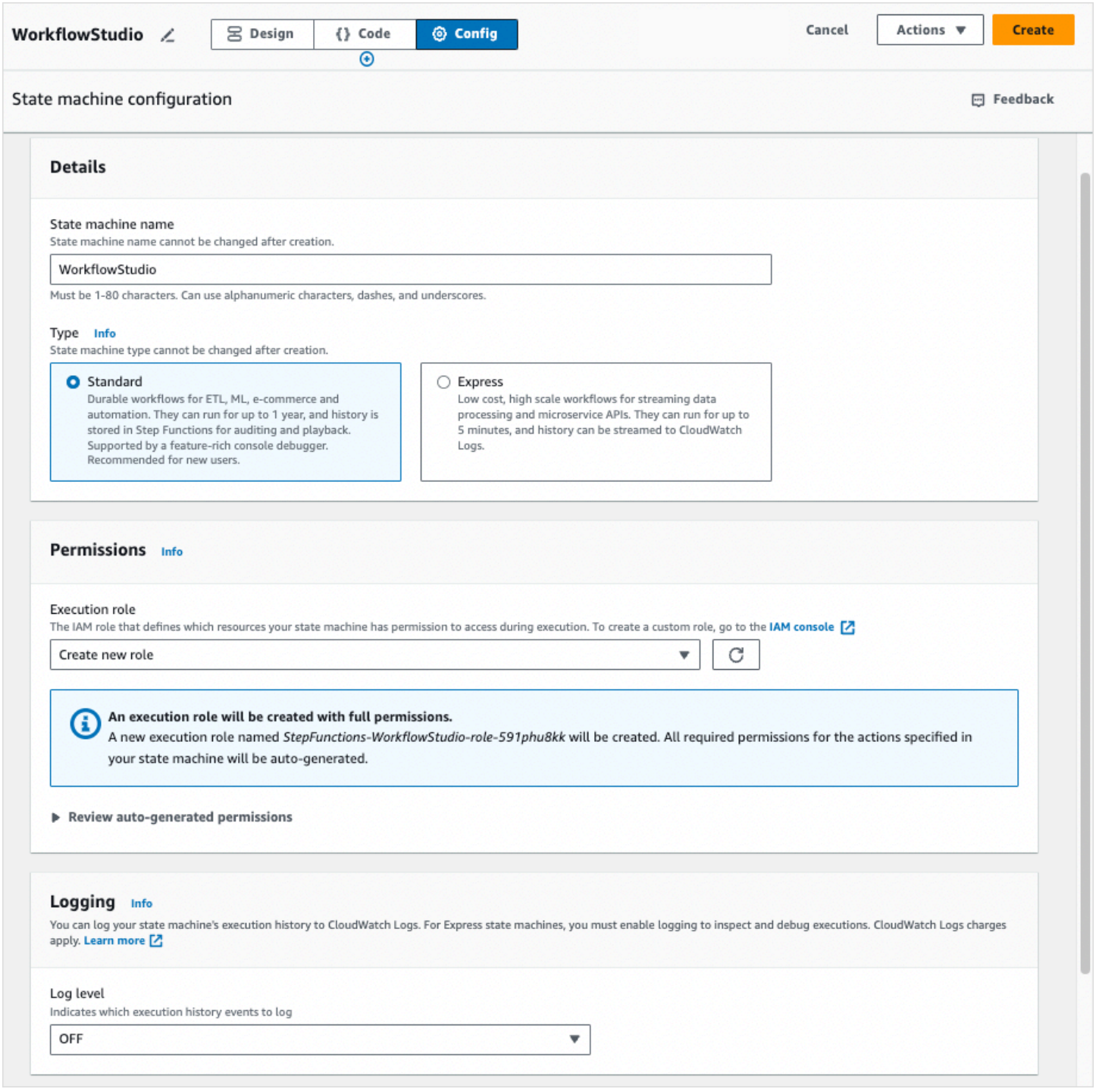
Le visualizzazioni grafiche consentono di vedere l'aspetto del flusso di lavoro in formato grafico. Quando scrivete le definizioni del flusso di lavoro in Workflow Studio, il riquadro Editor del codice di visualizzazione del grafico visualizza un grafico in tempo reale del flusso di lavoro. Quando riordinate, eliminate o duplicate uno stato nel riquadro di visualizzazione del grafico, la definizione del flusso di lavoro nell'editor di codice viene aggiornata automaticamente. Allo stesso modo, quando aggiorni le definizioni del flusso di lavoro, riordini, elimini o aggiungi uno stato nell'editor di codice, la visualizzazione viene aggiornata automaticamente.

Se il codice JSON nella definizione ASL del flusso di lavoro non è valido, il riquadro di visualizzazione del grafico sospende il rendering e visualizza un messaggio di stato nella parte inferiore del riquadro.

modalità Config

La modalità Config di Workflow Studio consente di gestire la configurazione delle macchine a stati. In questa modalità, è possibile specificare dettagli, come il nome e il tipo della macchina a stati, le autorizzazioni IAM e la configurazione di registrazione per la macchina a stati. Altre configurazioni

aggiuntive che è possibile specificare in questa modalità includono l'abilitazione del AWS X-Ray tracciamento e della pubblicazione di una versione quando si crea la macchina a stati. Dopo aver creato la macchina a stati, è possibile modificare tutte le opzioni di configurazione della macchina a stati tranne il nome e il tipo della macchina a stati. L'immagine seguente mostra alcune delle configurazioni che è possibile specificare in modalità Config.



Gestisci la configurazione delle macchine a stati

Per gestire la configurazione della macchina a stati, procedi come segue:

1. Immettete un nome per la vostra macchina a stati nella casella Nome macchina a stati.

> **Tip**
>
> In alternativa, scegli l'icona di modifica accanto al nome della macchina a stati predefinita di MyStateMachine. Quindi, in Configurazione della macchina a stati, specifica un nome.

> **Important**
>
> Non è possibile modificare il nome della macchina a stati dopo averla creata.

2. In Tipo, scegli un tipo di macchina a stati Standard o Express. Per informazioni sui tipi di macchine a stati, vedereFlussi di lavoro Standard e Express.

> **Important**
>
> Non è possibile modificare il tipo di macchina a stati dopo aver creato la macchina a stati.

3. In Autorizzazioni, seleziona il ruolo IAM da utilizzare come ruolo di esecuzione per la macchina a stati.
   - Crea nuovo ruolo (consigliato): se si seleziona questa opzione, Step Functions crea automaticamente un ruolo di esecuzione per le macchine a stati con i privilegi minimi richiesti quando si creano le macchine a stati. Questi ruoli IAM generati automaticamente sono validi per la creazione della macchina Regione AWS a stati.

> **Tip**
>
> Per rivedere le autorizzazioni che Step Functions genererà automaticamente per la tua macchina a stati, scegli Rivedi le autorizzazioni generate automaticamente.

> **Note**
>
> Se elimini il ruolo IAM creato da Step Functions, Step Functions non può ricrearlo in un secondo momento. Allo stesso modo, se modifichi il ruolo (ad esempio, rimuovendo

Step Functions dai principi nella policy IAM), Step Functions non può ripristinare le impostazioni originali in un secondo momento.

- Scegli un ruolo esistente: crea il tuo ruolo IAM per la macchina a stati, quindi sceglilo tra le opzioni elencate di seguito Scegli un ruolo esistente. Assicurati che la politica del ruolo includa le autorizzazioni che vorresti che la macchina a stati assumesse.

  Per informazioni sulla creazione di una policy IAM, consulta Creazione di policy IAM nella Guida per l'utente di IAM.

- Inserisci un ruolo ARN: specifica l'Amazon Resource Name (ARN) di un ruolo IAM esistente da utilizzare per questa macchina a stati. Ad esempio, `arn:aws:iam::123456789012:role/service-role/StepFunctions-WorkflowStudio-role-777f4027`.

4. In Logging, imposta il livello di registro per la tua macchina a stati. Step Functions registra gli eventi della cronologia di esecuzione in base alla selezione effettuata. È possibile selezionare una delle seguenti opzioni:

   - TUTTI: vengono registrati tutti i tipi di eventi.
   - ERRORE: vengono registrati tutti i tipi di eventi di errore, ad esempio e TaskFailed . ExecutionFailed
   - FATALE: vengono registrati tutti i tipi di eventi di errore irreversibile, ad esempio e. ExecutionAborted ExecutionFailed
   - OFF: non viene registrato alcun tipo di evento.

   Per ulteriori informazioni sui livelli di registro, vedereLivelli di log.

5. In Configurazione aggiuntiva, imposta una o più delle seguenti configurazioni opzionali:

   - Abilita X-Ray il tracciamento: scegli questa casella di controllo a Step Functions a cui inviare tracce X-Ray per le esecuzioni di macchine a stati, anche quando un ID di traccia non viene passato da un servizio upstream. Per ulteriori informazioni, consulta AWS X-Ray e Step Functions.
   - Pubblica versione al momento della creazione: una versione è un'istantanea numerata e immutabile di una macchina a stati che è possibile eseguire. Seleziona questa casella di controllo per pubblicare una versione della tua macchina a stati durante la creazione della macchina a stati. Step Functions pubblica la versione 1 come prima revisione della macchina a stati.

     Per ulteriori informazioni sulle versioni, consulta Versioni di macchine a stati.

- Aggiungi nuovo tag: scegli questa casella per aggiungere tag alla tua macchina a stati. L'aggiunta di tag può aiutarti a monitorare e gestire i costi associati alle tue risorse e a fornire una maggiore sicurezza nelle tue politiche IAM. Per ulteriori informazioni sui tag, consulta Inserimento di tag in Step Functions.

6. Scegli Crea.
7. Nella finestra di dialogo Conferma creazione del ruolo, scegli Conferma per continuare.

   Puoi anche scegliere Visualizza la configurazione del ruolo per tornare alla modalità Config.

Tasti di scelta rapida

Workflow Studio supporta le seguenti scelte rapide da tastiera:

Tasto di scelta rapida	Funzione
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

Utilizzo di Workflow Studio

Impara a creare, modificare ed eseguire flussi di lavoro utilizzando Step Functions Workflow Studio. Quando il flusso di lavoro è pronto, puoi esportarlo. Puoi anche utilizzare Workflow Studio per la prototipazione rapida.

In questo argomento

- Crea un flusso di lavoro
- Progetta un flusso di lavoro
- Esegui il tuo flusso di lavoro
- Modifica il tuo flusso di lavoro
- Esporta il flusso di lavoro
- Crea il prototipo del tuo flusso di lavoro

Crea un flusso di lavoro

In Workflow Studio, puoi scegliere un modello iniziale o scegliere un modello vuoto per creare un flusso di lavoro da zero. Per i modelli vuoti, puoi utilizzare la modalità Progettazione o Codice per creare il tuo flusso di lavoro.

Un modello iniziale è un progetto di ready-to-run esempio che crea automaticamente il prototipo e la definizione del flusso di lavoro e distribuisce tutte le AWS risorse correlate di cui il progetto ha bisogno per il progetto. Account AWSÈ possibile utilizzare questi modelli iniziali per distribuirli ed eseguirli così come sono oppure utilizzare i prototipi del flusso di lavoro per svilupparli. Per ulteriori informazioni sui modelli iniziali, consulta. Progetti di esempio per Step Functions

Crea un flusso di lavoro utilizzando i modelli iniziali

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Nella finestra di dialogo Scegli un modello, effettuate una delle seguenti operazioni per scegliere un progetto di esempio, ad esempio il progetto di esempio Task Timer:
   - Digitate **`Task Timer`** nella casella Cerca per parola chiave, quindi scegliete Task Timer dai risultati della ricerca restituiti.
   - Sfoglia i progetti di esempio elencati in Tutti nel riquadro a destra, quindi scegli Task Timer.
3. Seleziona Successivo per continuare.

4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.
5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro. InModalità di progettazione, trascina e rilascia gli stati da Stati del browser per continuare a creare il prototipo del tuo flusso di lavoro. Oppure passa alla sezione Modalità codice per aggiornare la definizione Amazon States Language (ASL) del tuo flusso di lavoro.

     > **Important**
     >
     > Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

   - Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

     > **Tip**
     >
     > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

     Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

     > **Note**
     >
     > La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Durante la distribuzione delle risorse, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

> **Important**
>
> Le tariffe standard si applicano per ogni servizio utilizzato nel modello. CloudFormation

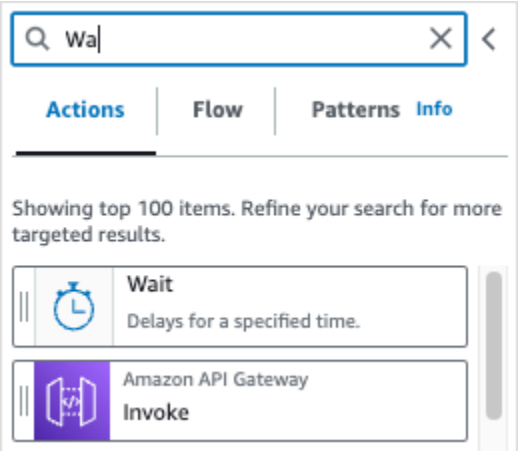
Crea un flusso di lavoro utilizzando un modello vuoto

1. Apri la console Step Functions.
2. Scegli Create state machine (Crea macchina a stati).
3. Nella finestra di dialogo Scegli un modello, seleziona Vuoto.
4. Scegli Seleziona. Questo apre Workflow Studio inModalità di progettazione.

   Ora puoi iniziare a progettare il tuo flusso di lavoro Modalità di progettazione o a scrivere la definizione del flusso di lavoro inModalità codice.
5. Scegli Config per gestire la configurazione del tuo flusso di lavoro in. modalità Config Ad esempio, fornisci un nome per il tuo flusso di lavoro e scegline il tipo.

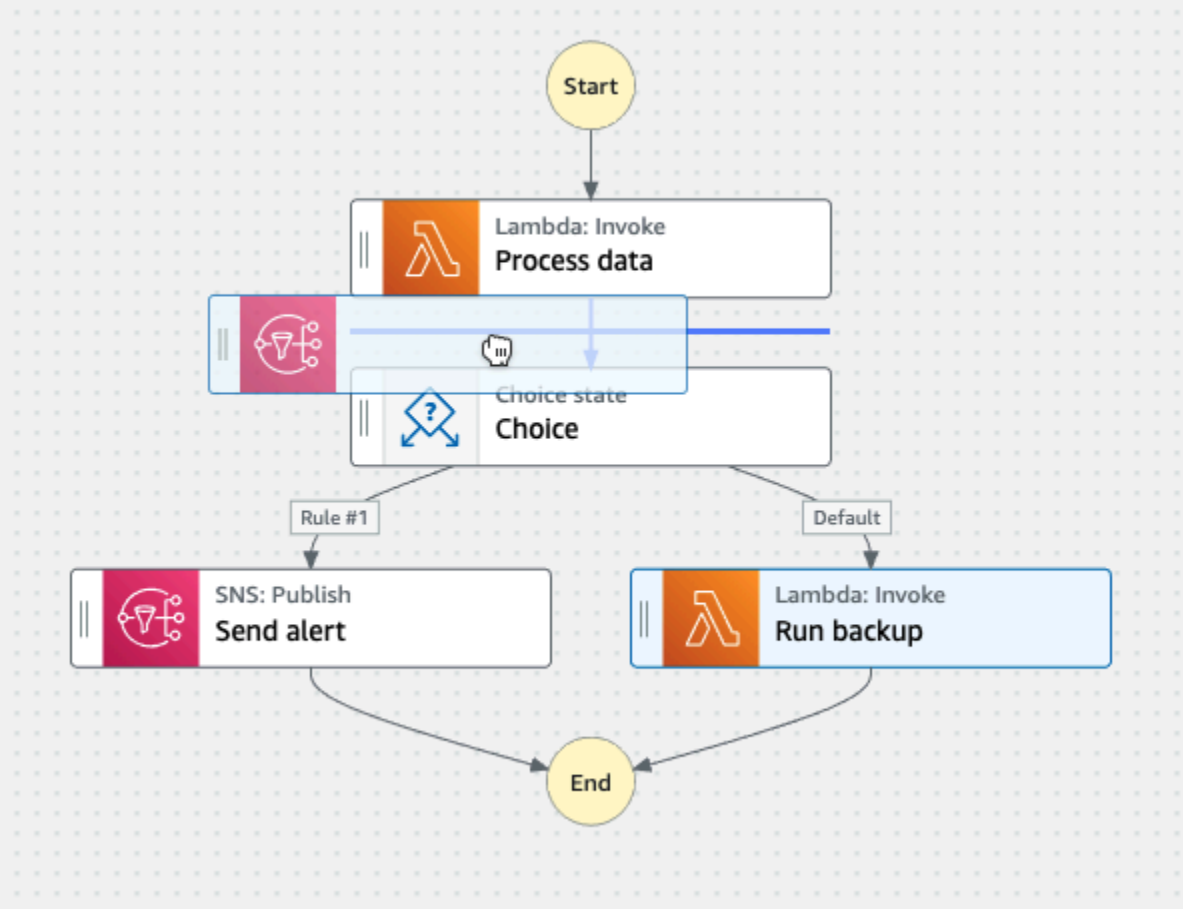
Progetta un flusso di lavoro

Se conosci il nome dello stato che desideri aggiungere, utilizza la casella di ricerca nella parte superiore Stati del browser per trovare lo stato nelle schede Azioni e Flusso diModalità di progettazione.

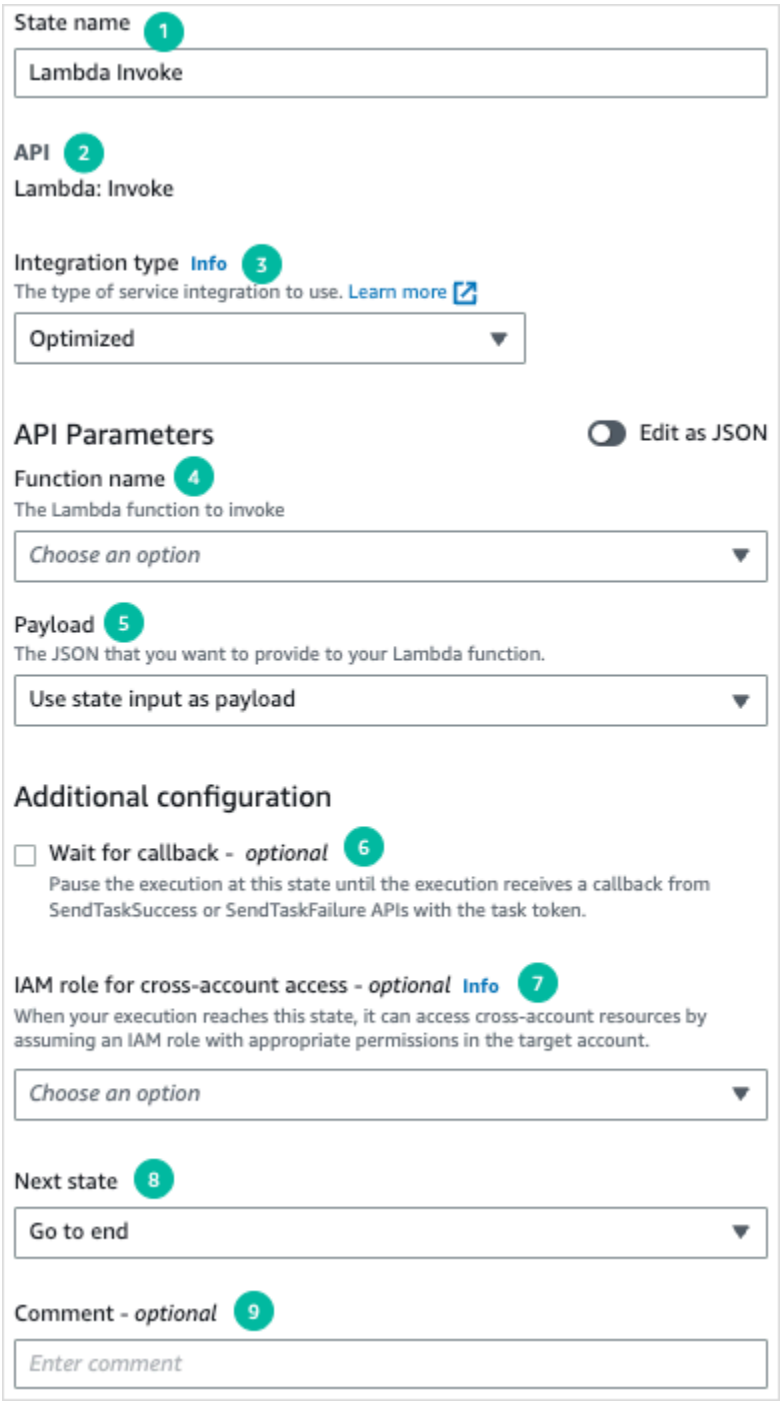


Altrimenti, scegli uno stato dal browser degli stati e trascinalo sull'area di disegno, posizionandolo dove preferisci nel tuo flusso di lavoro. Puoi anche riordinare gli stati del flusso di lavoro trascinandoli

in una posizione diversa nel flusso di lavoro. Quando trascini uno stato sull'area di disegno, appare una linea ovunque tu possa rilasciarlo nel flusso di lavoro. Dopo che uno stato è stato inserito nell'area di disegno, il relativo codice viene generato automaticamente e aggiunto nella definizione del flusso di lavoro. Per visualizzare la definizione, attiva l'interruttore Definition nel pannello Inspector. Per apportare modifiche alla definizione del flusso di lavoro, scegli Modalità codice quello che offre un editor di codice integrato.



Dopo aver rilasciato uno stato sull'area di disegno, puoi configurarlo nel ✓Inspector pannello a destra. Questo pannello contiene le schede Configurazione, Input, Output e Gestione degli errori per ciascuno degli stati o delle azioni API che inserite nell'area di disegno. Puoi configurare gli stati da includere nei flussi di lavoro nella scheda Configurazione. Ad esempio, la scheda Configurazione per l'azione Lambda Invoke API è composta dalle seguenti opzioni:



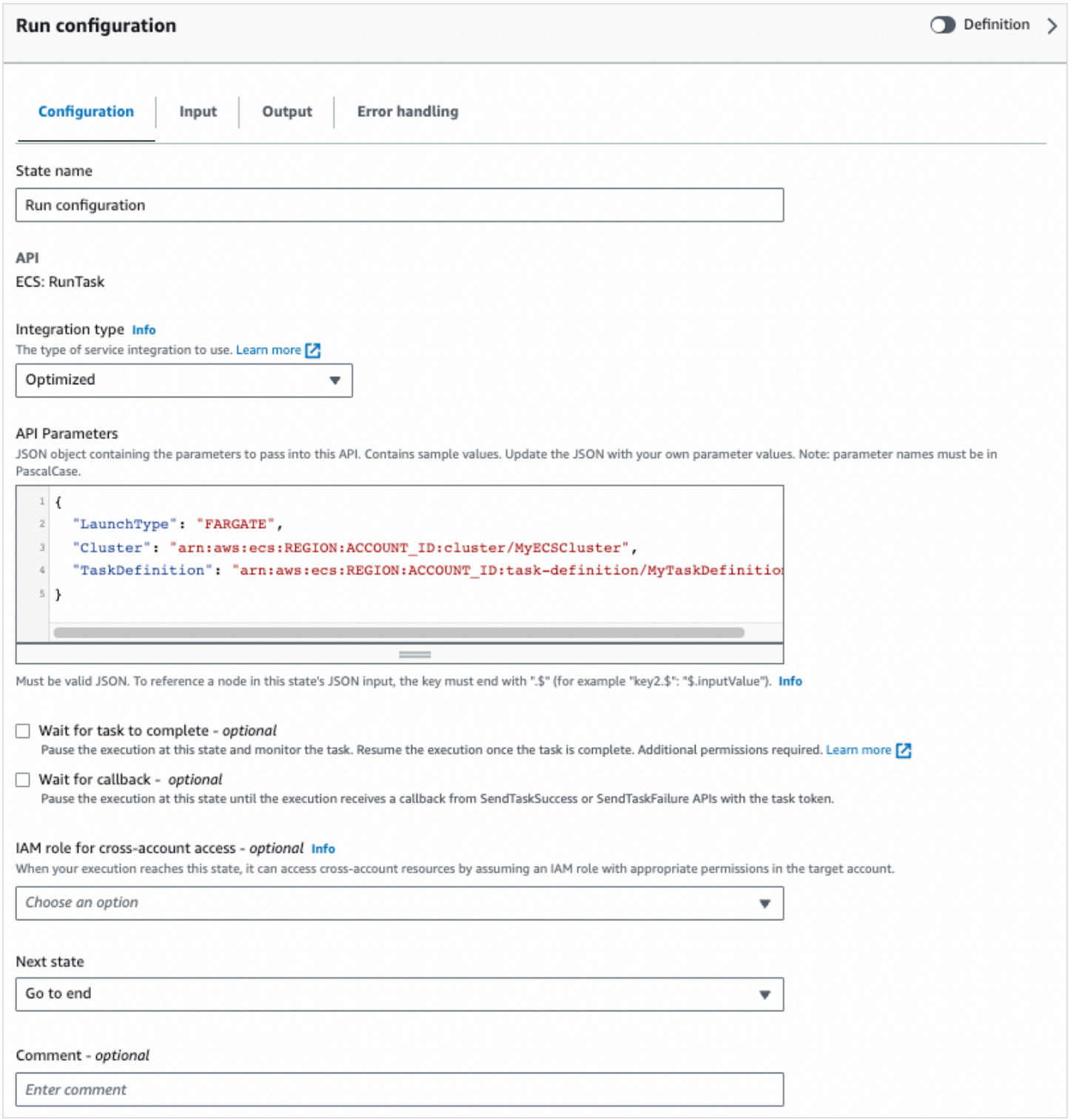
1. Il nome dello stato identifica lo stato. Puoi usare il tuo nome o accettare il nome generato di default.
2. L'API mostra l'azione API utilizzata dallo stato.
3. L'elenco a discesa Tipo di integrazione fornisce opzioni per scegliere il tipo di integrazioni di servizi disponibili in Step Functions. Il tipo di integrazione scelto viene utilizzato per richiamare azioni API specifiche del flusso di Servizio AWS lavoro.

4. Il nome della funzione fornisce opzioni per:
   - Inserisci il nome di una funzione: puoi inserire il nome della tua funzione o il relativo ARN.
   - Ottieni il nome della funzione in fase di esecuzione dall'input dello stato: puoi usare questa opzione per ottenere dinamicamente il nome della funzione dall'input dello stato in base al percorso specificato.
   - Seleziona il nome della funzione: puoi selezionare direttamente tra le funzioni disponibili nel tuo account e nella tua regione.
5. Il Payload consente di selezionare tra le seguenti opzioni:
   - Usa l'input dello stato come payload: puoi usare questa opzione per passare l'input dello stato come payload fornito alla tua funzione Lambda.
   - Inserisci il tuo payload: puoi usare questa opzione per costruire un oggetto JSON da passare come payload alla tua funzione Lambda. Questo JSON può includere sia valori statici che valori selezionati dall'input dello stato.
   - Nessun payload: puoi usare questa opzione se non vuoi passare alcun payload alla tua funzione Lambda.
6. (Facoltativo) In alcuni stati è possibile selezionare Attendi il completamento dell'attività o Attendi la richiamata. Se disponibili, queste opzioni selezionano uno dei seguenti modelli di integrazione dei servizi:
   - Nessuna opzione selezionata: Step Functions utilizzerà il modello di Richiesta e risposta integrazione. Step Functions attenderà una risposta HTTP e poi passerà allo stato successivo. Step Functions non aspetterà il completamento di un lavoro. Quando non sono disponibili opzioni, lo stato utilizzerà questo modello.
   - Attendi il completamento dell'operazione: Step Functions utilizzerà il modello di Esegui un processo (.sync) integrazione.
   - Attendi la richiamata: Step Functions utilizzerà il modello di Attendere un callback con il token dell'attività integrazione.
7. (Facoltativo) Per accedere a risorse configurate in modo diverso Account AWS all'interno dei flussi di lavoro, Step Functions fornisce l'accesso su più account. Il ruolo IAM per l'accesso tra account diversi offre opzioni per:
   - Fornisci l'ARN del ruolo IAM: specifica il ruolo IAM che contiene le autorizzazioni di accesso alle risorse appropriate. Queste risorse sono disponibili in un account di destinazione, che è un account Account AWS a cui si effettuano chiamate tra account.

- Ottieni l'ARN del ruolo IAM in fase di esecuzione dall'input dello stato: specifica un percorso di riferimento a una coppia chiave-valore esistente nell'input JSON dello stato che contiene il ruolo IAM.

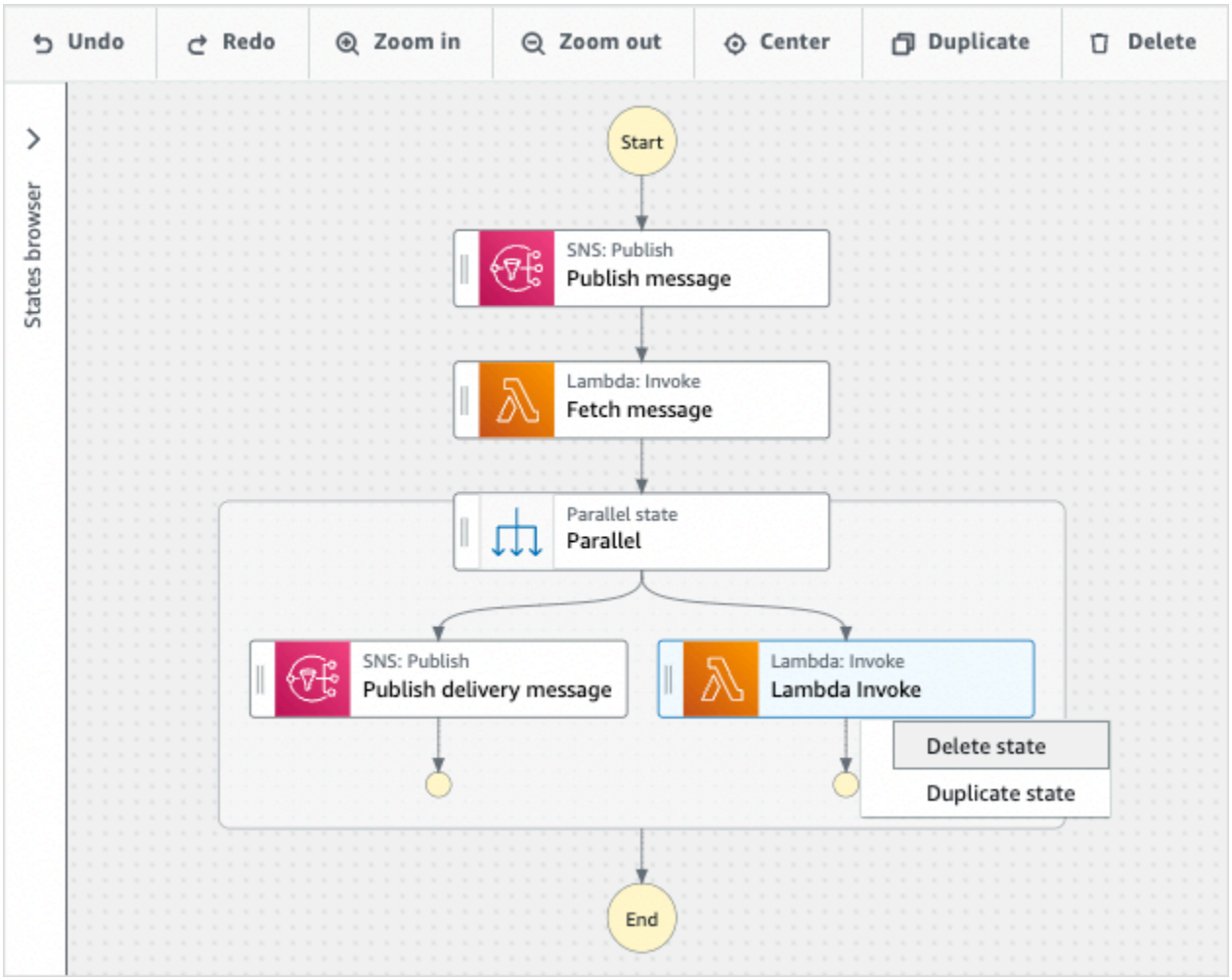
8. Lo stato successivo ti consente di selezionare lo stato successivo a cui desideri passare.
9. (Facoltativo) Il campo Commento può essere utilizzato per aggiungere un commento personalizzato. Non influirà sul flusso di lavoro, ma può essere utilizzato per annotare il flusso di lavoro.

Alcuni stati avranno opzioni di configurazione più generiche. Ad esempio, la configurazione `RunTask` dello stato di Amazon ECS contiene un `API Parameters` campo popolato con valori segnaposto.



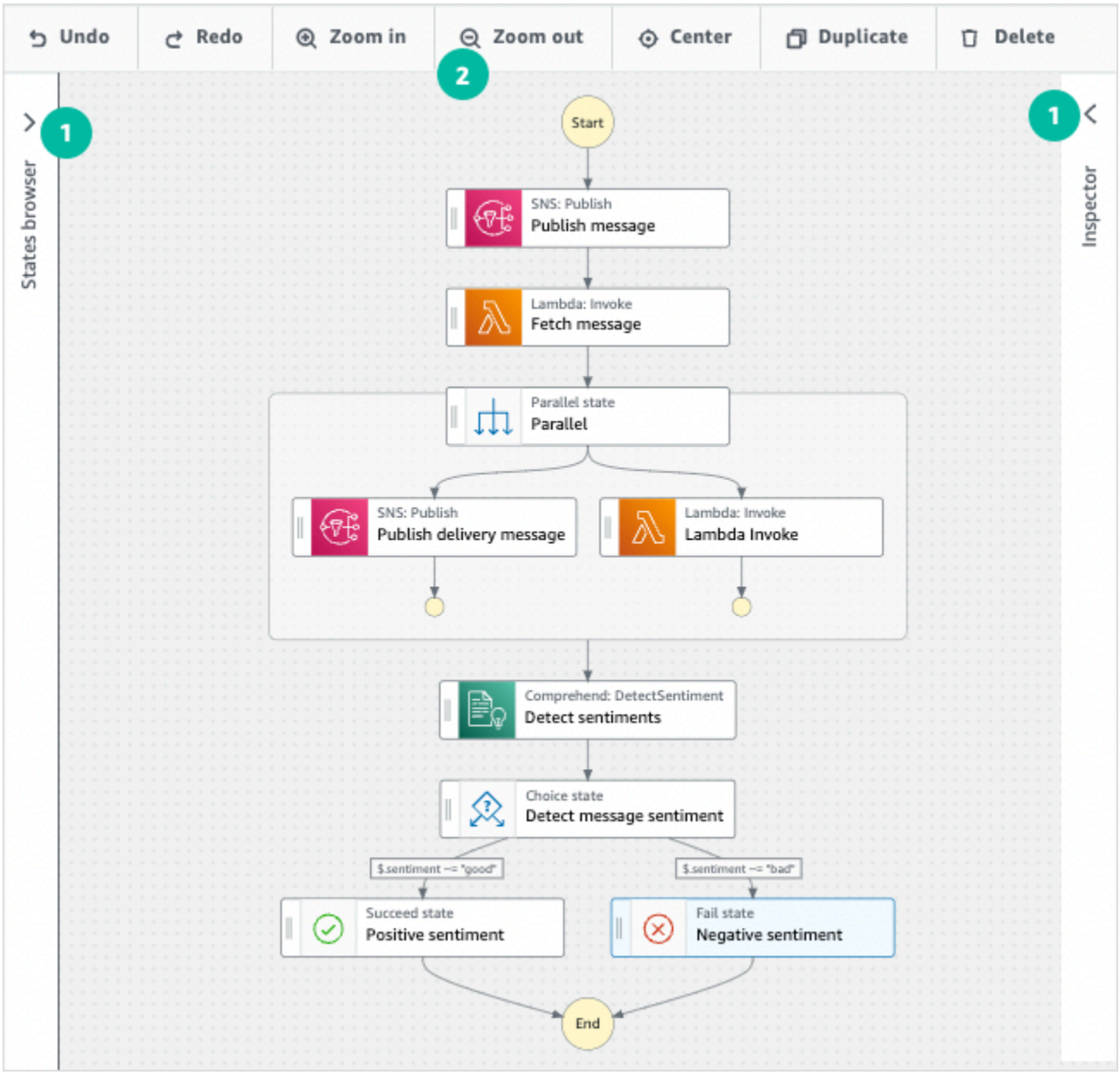
Per questi stati, puoi sostituire i valori segnaposto con configurazioni adatte alle tue esigenze.

Per eliminare uno stato, puoi usare backspace, fare clic con il pulsante destro del mouse e scegliere Elimina stato oppure scegliere Elimina nella barra degli strumenti Progettazione.



Man mano che il flusso di lavoro cresce, potrebbe non rientrare nell'area di lavoro. È possibile:

1. Utilizza i controlli nei pannelli laterali per ridimensionare o chiudere i pannelli.
2. Utilizza i controlli della barra degli strumenti Design nella parte superiore Canvas per ingrandire o ridurre il grafico del flusso di lavoro.



Esegui il tuo flusso di lavoro

Dopo aver creato o modificato il flusso di lavoro con Workflow Studio, puoi eseguirlo e visualizzarne l'esecuzione nella console Step Functions.

Per eseguire un flusso di lavoro in Workflow Studio

1. In modalità Progettazione, Codice o Config, scegliete Esegui.

   La finestra di dialogo Avvia esecuzione si apre in una nuova scheda.
2. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

   > **Note**
   >
   > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.
3. Selezionare Start execution (Avvia esecuzione).
4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

   Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

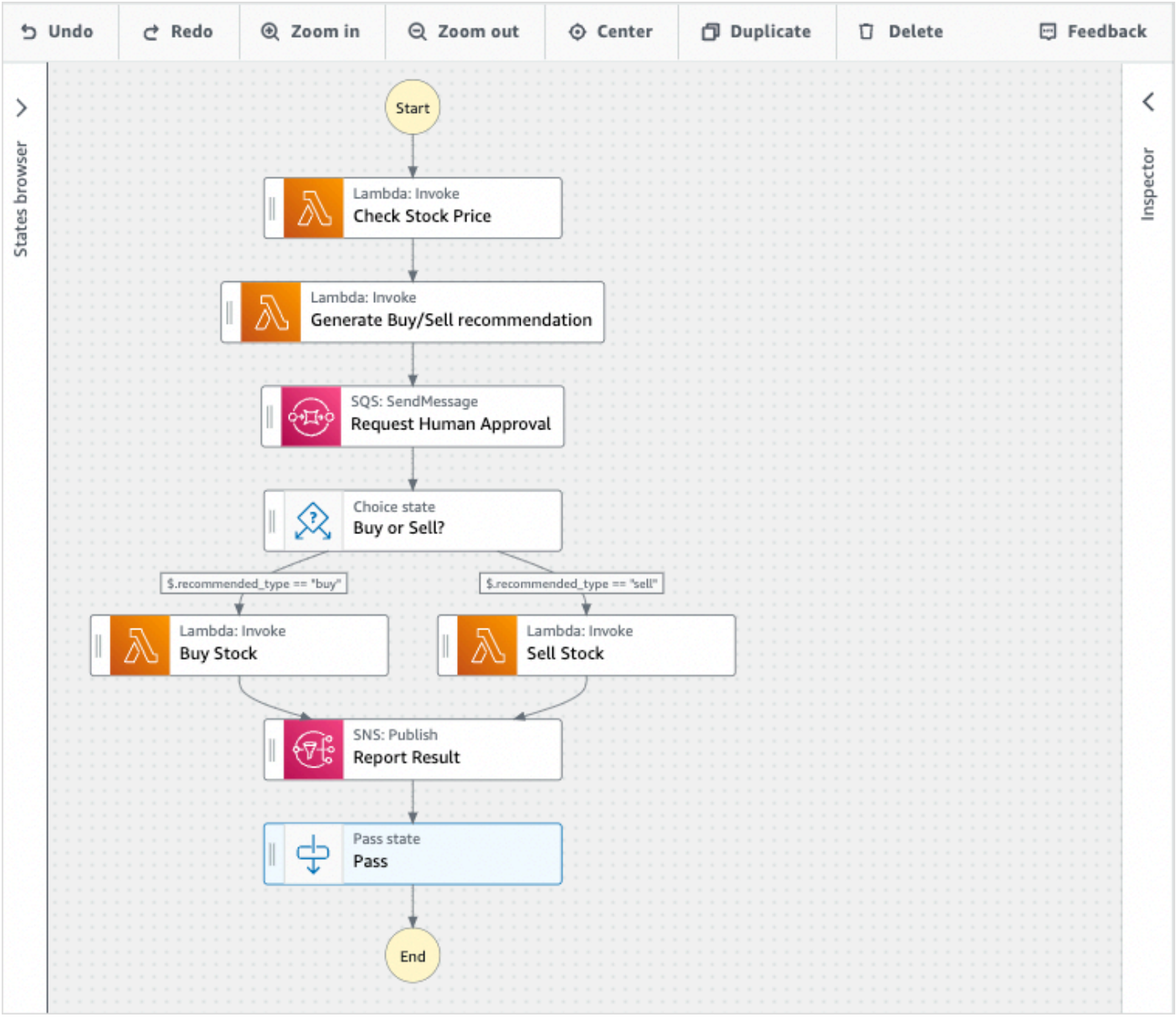
Modifica il tuo flusso di lavoro

Puoi modificare visivamente un flusso di lavoro esistente in Workflow Studio. Modalità di progettazione È inoltre possibile modificare la definizione del flusso Modalità codice di lavoro in Workflow Studio.

Per modificare un flusso di lavoro esistente:

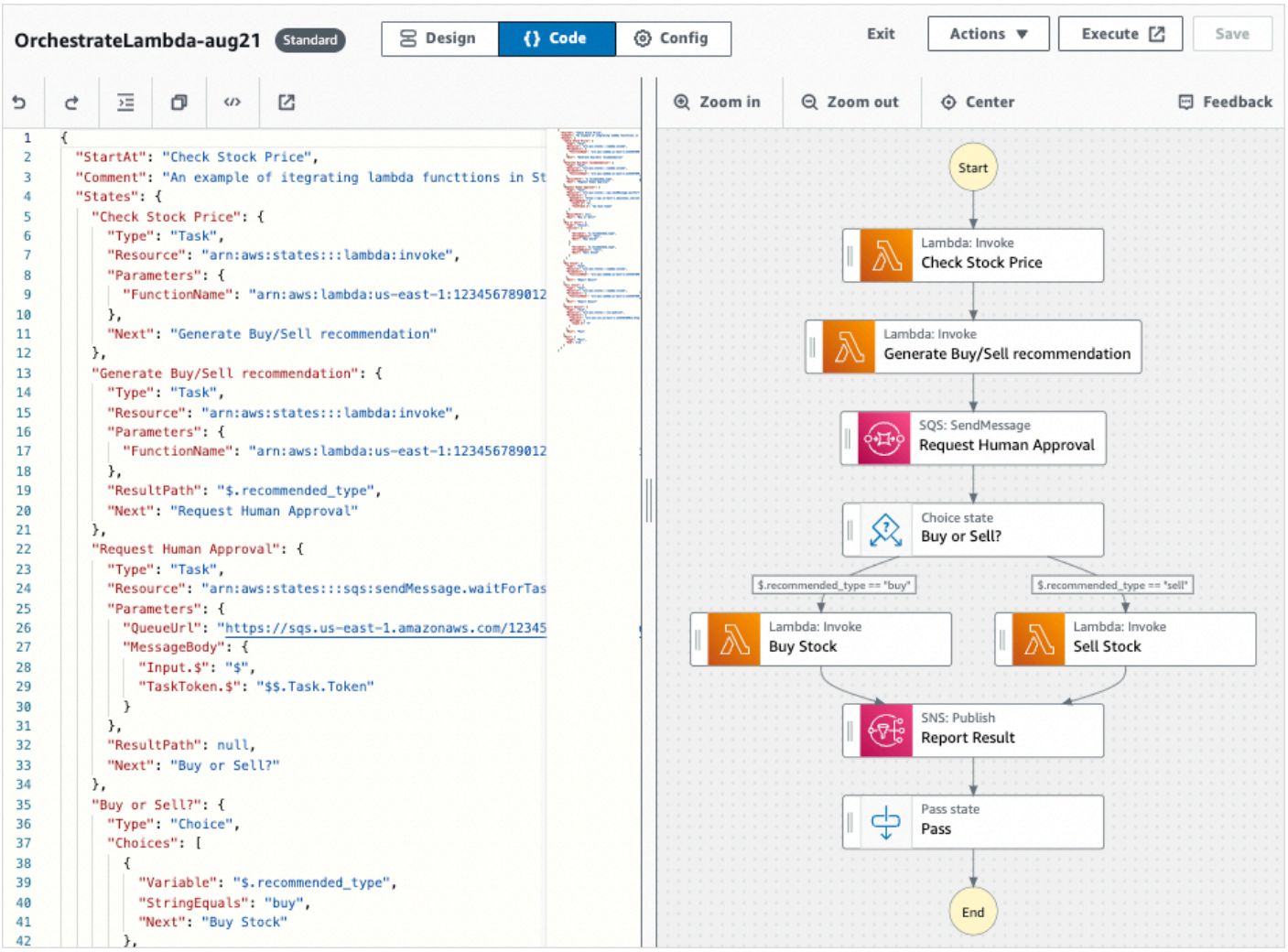
1. Apri la console Step Functions.
2. Nella pagina Macchine a stati, scegli il flusso di lavoro che desideri modificare.
3. Nella pagina dei dettagli della macchina a stati, scegli Modifica.

4. Il flusso di lavoro viene aperto in modalità Progettazione di Workflow Studio. Modificate il flusso di lavoro come richiesto.



> **Note**
>
> Se riscontri errori nel flusso di lavoro, devi correggerli in modalità Progettazione. Non puoi passare alla modalità Codice o Config se nel flusso di lavoro sono presenti errori.

5. (Facoltativo) Scegliete il pulsante Codice per visualizzare o modificare la definizione del flusso di lavoro in Workflow Studio.



6. Al termine, scegli Salva per salvare il flusso di lavoro aggiornato.
7. (Facoltativo) Per eseguire il flusso di lavoro aggiornato, scegli Esegui. La finestra di dialogo Avvia esecuzione si apre in una nuova scheda.

Esporta il flusso di lavoro

Puoi esportare la definizione del flusso di lavoro Amazon States Language (ASL) e il grafico del flusso di lavoro:

1. Scegli il tuo flusso di lavoro nella console Step Functions.
2. Nella pagina dei dettagli della macchina a stati, scegli Modifica.
3. (Facoltativo) Il flusso di lavoro viene aperto in modalità Progettazione di Workflow Studio. Modifica il flusso di lavoro in modalità Progettazione o passa alla modalità Codice.
4. Scegli il pulsante a discesa Azioni, quindi esegui una o entrambe le seguenti operazioni:

- Per esportare il grafico del flusso di lavoro in un file SVG o PNG, in Esporta grafico, seleziona il formato desiderato.
- Per esportare la definizione del flusso di lavoro come file JSON o YAML, in Esporta definizione, selezionate il formato desiderato.

Crea il prototipo del tuo flusso di lavoro

È possibile utilizzare Workflow Studio per creare prototipi di nuovi flussi di lavoro che contengono risorse segnaposto. Puoi anche creare i tuoi flussi di lavoro utilizzando Workflow Studio in. Application Composer Per creare un prototipo:

1. Accedi alla console Step Functions.
2. Scegli Create state machine (Crea macchina a stati).
3. Nella finestra di dialogo Scegli un modello, seleziona Vuoto.
4. Scegli Seleziona. Questo apre Workflow Studio inModalità di progettazione.
5. Si apre la modalità Progettazione di Workflow Studio. Progetta il tuo flusso di lavoro in Workflow Studio. Per includere risorse segnaposto:
   a. Scegli lo stato per il quale desideri includere una risorsa segnaposto, quindi in Configurazione:
      - Per gli stati Lambda Invoke, scegli Nome funzione, quindi scegli Inserisci il nome della funzione. Puoi anche inserire un nome personalizzato per la tua funzione.
      - Per gli stati di invio dei messaggi di Amazon SQS, scegli Queue URL, quindi scegli Enter queue URL. Inserisci un URL di coda segnaposto.
      - Per gli stati di Amazon SNS Publish, da Argomento, scegli un argomento ARN.
      - Per tutti gli altri stati elencati in Azioni, puoi utilizzare la configurazione predefinita.

      > **Note**
      >
      > Se riscontrate errori nel flusso di lavoro, dovete correggerli in modalità Progettazione. Non puoi passare alla modalità Codice o Config se nel flusso di lavoro sono presenti errori.

   b. (Facoltativo) Per visualizzare la definizione ASL generata automaticamente del flusso di lavoro, scegli Definizione.
   c. (Facoltativo) Per aggiornare la definizione del flusso di lavoro in Workflow Studio, scegliete il pulsante Codice.

> **Note**
>
> Se notate errori nella definizione del flusso di lavoro, dovete correggerli in modalità Codice. Non puoi passare alla modalità Design o Config se esistono errori nella definizione del flusso di lavoro.

6. (Facoltativo) Per modificare il nome della macchina a stati, scegliete l'icona di modifica accanto al nome della macchina a stati predefinita di MyStateMachinee specificate un nome nella casella Nome macchina a stati.

   È inoltre possibile passare modalità Config alla modifica del nome predefinito della macchina a stati.

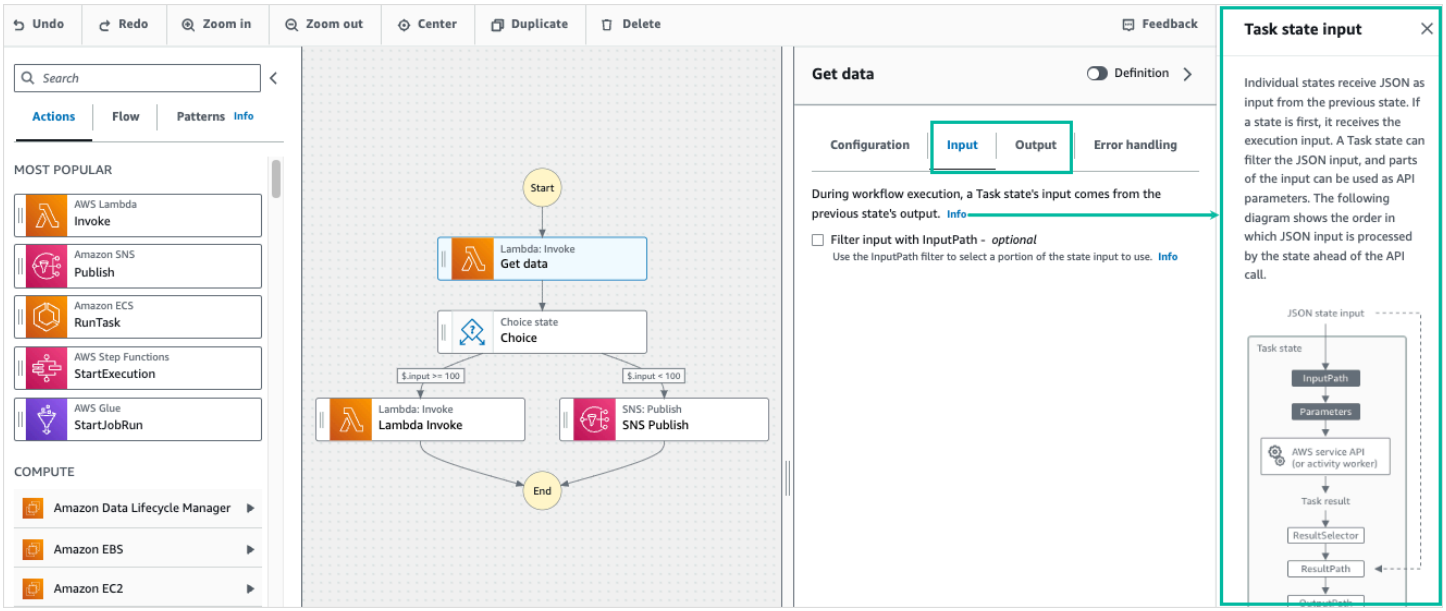
7. Specificate le impostazioni del flusso di lavoro, ad esempio il tipo di macchina a stati e il relativo ruolo di esecuzione.
8. Scegli Crea.

Ora hai creato un nuovo flusso di lavoro con risorse segnaposto che possono essere utilizzate per creare prototipi. Puoi esportare la definizione del flusso di lavoro e il grafico del flusso di lavoro.

- Per esportare la definizione del flusso di lavoro come file JSON o YAML, in modalità Progettazione o Codice, scegli il pulsante a discesa Azioni. Quindi, in Definizione di esportazione, seleziona il formato che desideri esportare. È possibile utilizzare questa definizione esportata come punto di partenza per lo sviluppo locale con. AWS Toolkit for Visual Studio Code
- Per esportare il grafico del flusso di lavoro in un file SVG o PNG, in modalità Progettazione o Codice, scegli il pulsante a discesa Azioni. Quindi, in Definizione di esportazione, selezionate il formato desiderato.

Configura ingressi e uscite per i tuoi stati

Ogni stato prende una decisione o esegue un'azione in base agli input ricevuti. Nella maggior parte dei casi, passa quindi l'output ad altri stati. In Workflow Studio, potete configurare il modo in cui uno stato filtra e manipola i dati di input e output nelle schede Input e Output del pannello. ✓Inspector Utilizzate i collegamenti Info per accedere alla guida contestuale per la configurazione di input e output.



Per informazioni dettagliate su come Step Functions elabora input e output, vedereElaborazione di input e output in Step Functions.

Configurare l'input in uno stato

Ogni stato riceve l'input dallo stato precedente come JSON. Se desideri filtrare l'input, puoi utilizzare il `InputPath` filtro nella scheda Input del ✓Inspector pannello. `InputPath`È una stringa, che inizia con$, che identifica un nodo JSON specifico. Questi sono chiamati percorsi di riferimento e seguono JsonPath la sintassi.

Per filtrare l'input:

- Scegli Filtra input con InputPath.
- Inserisci un valore valido JsonPathper il `InputPath` filtro. Ad esempio, **`$.data`**.

Il `InputPath` filtro verrà aggiunto al flusso di lavoro.

Example Esempio 1: utilizzo del InputPath filtro in Workflow Studio

Supponiamo che l'input del tuo stato includa i seguenti dati JSON.

```
{
  "comment": "Example for InputPath",
  "dataset1": {
    "val1": 1,
    "val2": 2,
    "val3": 3
  },
  "dataset2": {
    "val1": "a",
    "val2": "b",
    "val3": "c"
  }
}
```

Per applicare il `InputPath` filtro, scegli Filtra input con InputPath, quindi inserisci un percorso di riferimento appropriato. Se inserite**`$.dataset2.val1`**, il seguente codice JSON viene passato come input allo stato.

```
{"a"}
```

Un percorso di riferimento può anche avere una selezione di valori. Se i dati a cui fai riferimento sono `{ "a": [1, 2, 3, 4] }` e applichi il percorso di riferimento `$.a[0:2]` come `InputPath` filtro, il risultato è il seguente.

```
[ 1, 2 ]
```

ParallelEseguire la mappatura, e gli stati di Pass flusso dispongono di un'opzione di filtro di input aggiuntiva nella scheda Input. `Parameters` Questo filtro ha effetto dopo il InputPath filtro e può essere utilizzato per creare un oggetto JSON personalizzato costituito da una o più coppie chiave-valore. I valori di ogni coppia possono essere valori statici, possono essere selezionati dall'input o possono essere selezionati da un percorso. Oggetto Context

> **Note**
>
> Per specificare che un parametro utilizza un percorso di riferimento per puntare a un nodo JSON nell'input, il nome del parametro deve terminare con `.$`.

Example Esempio 2: creazione di un input JSON personalizzato per Parallel state

Supponiamo che i seguenti dati JSON siano l'input per uno stato parallelo.

```
{
  "comment": "Example for Parameters",
  "product": {
    "details": {
      "color": "blue",
      "size": "small",
      "material": "cotton"
    },
    "availability": "in stock",
    "sku": "2317",
    "cost": "$23"
  }
}
```

Per selezionare parte di questo input e passare ulteriori coppie chiave-valore con un valore statico, puoi specificare quanto segue nel campo Parametri, nella scheda Input dello stato Parallelo.

```
{
 "comment": "Selecting what I care about.",
 "MyDetails": {
    "size.$": "$.product.details.size",
    "exists.$": "$.product.availability",
    "StaticValue": "foo"
    }
 }
```

Il risultato saranno i seguenti dati JSON.

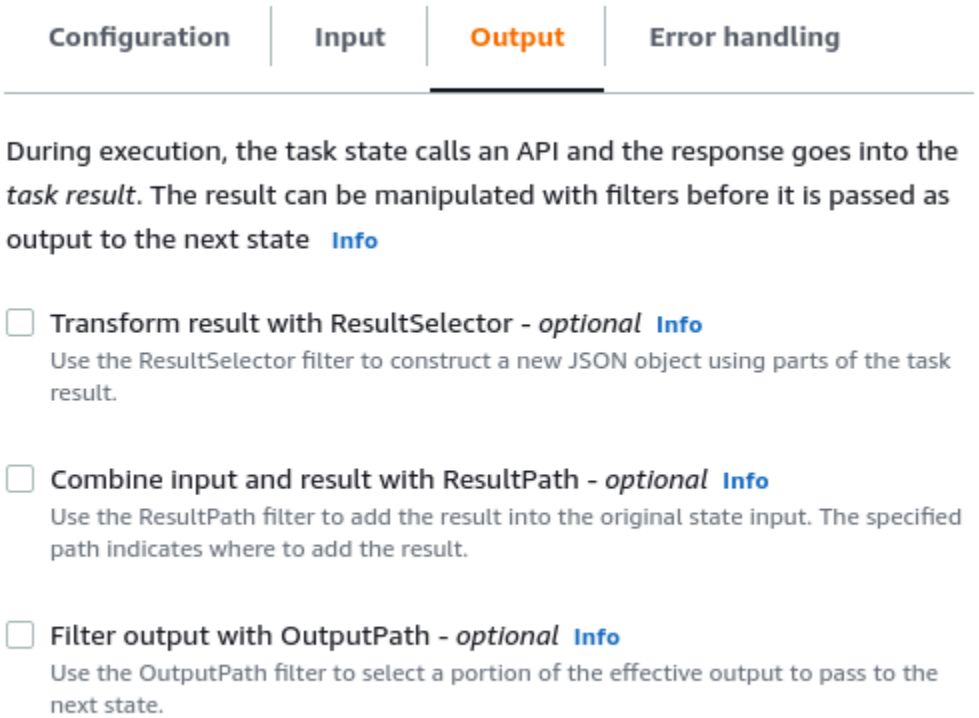
```
{
  "comment": "Selecting what I care about.",
  "MyDetails": {
```

```
    "size": "small",
    "exists": "in stock",
    "StaticValue": "foo"
  }
}
```

Configura l'output di uno stato

Ogni stato produce un output JSON che può essere filtrato prima di passare allo stato successivo. Sono disponibili diversi filtri, ognuno dei quali influisce sull'output in modo diverso. I filtri di output disponibili per ogni stato sono elencati nella scheda Output del pannello Inspector. Per Attività gli stati, tutti i filtri di output selezionati vengono elaborati nel seguente ordine:

1. `ResultSelector`: utilizzate questo filtro per manipolare il risultato dello stato. È possibile creare un nuovo oggetto JSON con parti del risultato.
2. `ResultPath`: utilizzate questo filtro per selezionare una combinazione dell'input dello stato e del risultato dell'attività da passare all'output.
3. `OutputPath`: Usa questo filtro per filtrare l'output JSON e scegliere quali informazioni del risultato verranno passate allo stato successivo.



Usa ResultSelector

`ResultSelector`è un filtro di output opzionale per i seguenti stati:

- Attivitàstati, che sono tutti gli stati elencati nella scheda Azioni diStati del browser.
- Eseguire la mappaturastati, nella scheda Flow del browser States.
- Parallelstates, nella scheda Flow del browser States.

`ResultSelector`può essere usato per costruire un oggetto JSON personalizzato costituito da una o più coppie chiave-valore. I valori di ogni coppia possono essere valori statici o selezionati dal risultato dello stato con un percorso.

> **Note**
>
> Per specificare che un parametro utilizza un percorso per fare riferimento a un nodo JSON nel risultato, il nome del parametro deve terminare con `.$`.

Example Esempio di utilizzo ResultSelector del filtro

In questo esempio, viene utilizzata `ResultSelector` per manipolare la risposta della chiamata API Amazon CreateCluster EMR per uno stato Amazon `CreateCluster` EMR. Di seguito è riportato il risultato della chiamata all'`CreateCluster`API Amazon EMR.

```
{
  "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"
  }
}
```

Per selezionare parte di queste informazioni e passare una coppia chiave-valore aggiuntiva con un valore statico, specifica quanto segue nel ResultSelectorcampo, nella scheda Output dello stato.

```
{
 "result": "found",
 "ClusterId.$": "$.output.ClusterId",
 "ResourceType.$": "$.resourceType"
 }
```

L'utilizzo `ResultSelector` produce il seguente risultato.

```
{
 "result": "found",
 "ClusterId": "AKIAIOSFODNN7EXAMPLE",
 "ResourceType": "elasticmapreduce"
}
```

Usa ResultPath

L'output di uno stato può essere una copia del suo input, il risultato che produce o una combinazione del suo input e del suo risultato. Utilizza `ResultPath` per controllare quale di queste combinazioni verrà passata all'output dello stato. Per ulteriori casi d'uso di`ResultPath`, vedereResultPath.

`ResultPath`è un filtro di output opzionale per i seguenti stati:

- Attivitàstati, che sono tutti gli stati elencati nella scheda Azioni del browser Stati.
- Eseguire la mappaturastati, nella scheda Flow del browser States.
- Parallelstates, nella scheda Flow del browser States.
- Passstates, nella scheda Flow del browser States.

`ResultPath`può essere usato per aggiungere il risultato all'input dello stato originale. Il percorso specificato indica dove aggiungere il risultato.

Example Esempio di utilizzo ResultPath del filtro

Supponiamo che quanto segue sia l'input di uno stato di Task.

```
{
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

Il risultato dello stato dell'attività è il seguente.

```
Hello, AWS Step Functions
```

È possibile aggiungere questo risultato all'input dello stato applicando `ResultPath` e immettendo un percorso di riferimento che indichi dove aggiungere il risultato, ad esempio`$.taskresult`:

Con ciò`ResultPath`, quanto segue è il JSON che viene passato come output dello stato.

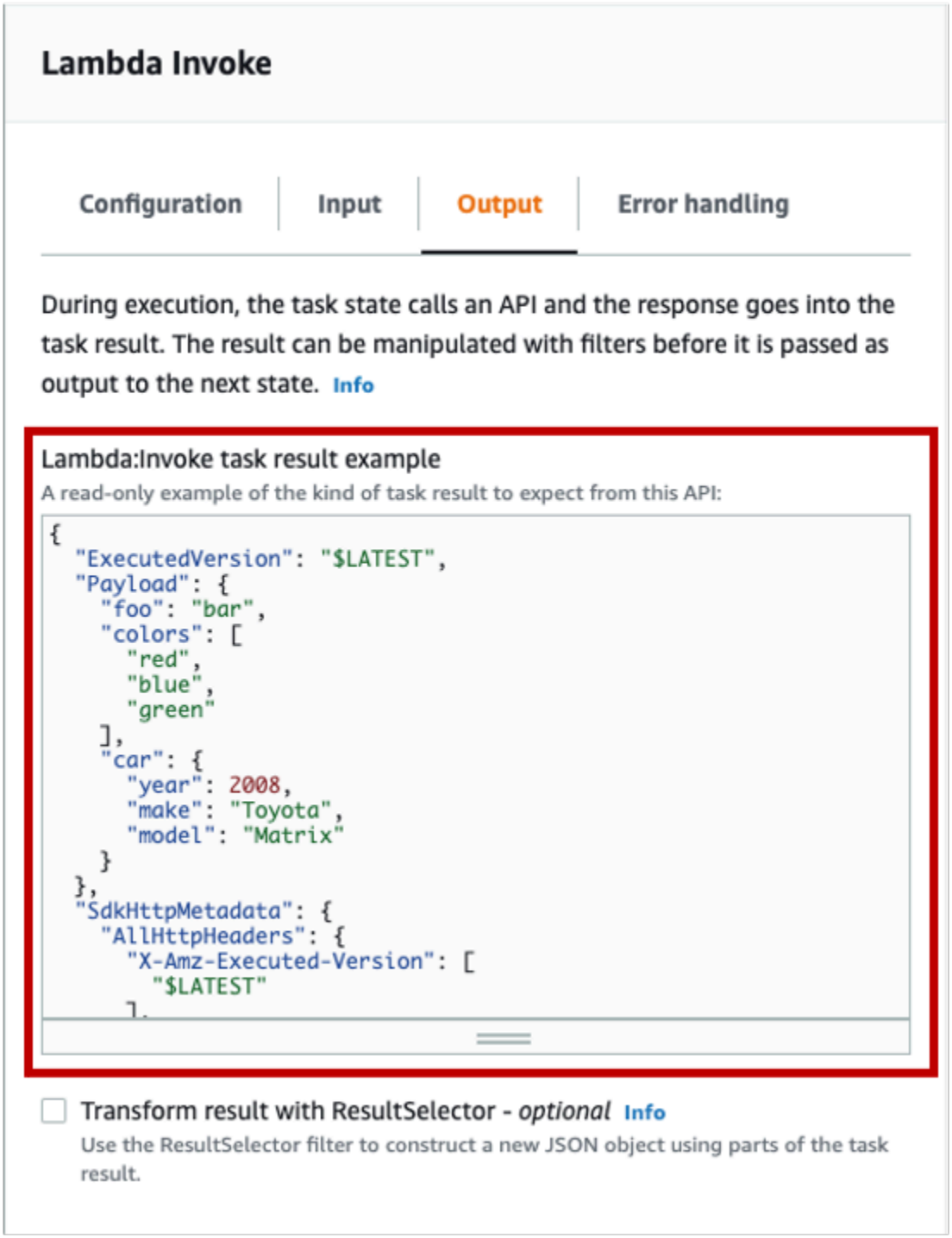
```
{
  "details": "Default example",
  "who": "AWS Step Functions",
  "taskresult": "Hello, AWS Step Functions!"
}
```

Usa OutputPath

Il `OutputPath` filtro ti consente di filtrare le informazioni indesiderate e di trasmettere solo la parte di JSON che ti interessa. `OutputPath`è una stringa, che inizia con$, che identifica i nodi all'interno del testo JSON.

Example Esempio di utilizzo del filtro OutputPath

Una chiamata all'API Lambda Invoke restituisce i metadati oltre al payload, che è il risultato della funzione Lambda. Un esempio della risposta di questa chiamata API è mostrato nella scheda Output dello stato.



Puoi usarlo `OutputPath` per filtrare i metadati aggiuntivi. Per impostazione predefinita, il valore del OutputPathfiltro per gli stati Lambda Invoke creati tramite Workflow Studio è. `$.Payload` Questo valore predefinito rimuove i metadati aggiuntivi e restituisce un output equivalente all'esecuzione diretta della funzione Lambda.

L'esempio di risultato dell'attività Lambda Invoke e il valore di `$.Payload` per il filtro Output passano i seguenti dati JSON come output.

```
{
 "foo": "bar",
 "colors": [
      "red",
      "blue",
      "green"
 ],
 "car": {
      "year": 2008,
      "make": "Toyota",
      "model": "Matrix"
 }
}
```

> **Note**
>
> Poiché il `OutputPath` filtro è l'ultimo filtro di output ad avere effetto, se si utilizzano filtri di output aggiuntivi come `ResultSelector` o`ResultPath`, è necessario modificare di conseguenza il valore predefinito di `$.Payload` per il `OutputPath` filtro.

Ruoli di esecuzione in Workflow Studio

Ogni macchina a Step Functions stati richiede un ruolo AWS Identity and Access Management (IAM) che concede alla macchina a stati l'autorizzazione a eseguire azioni su risorse Servizi AWS e o chiamare API di terze parti. Questo ruolo è chiamato ruolo di esecuzione. Questo ruolo deve contenere IAM politiche per ogni azione, ad esempio politiche che consentano alla macchina a stati di richiamare una AWS Lambda funzione, eseguire un AWS Batch processo o chiamare l'API Stripe. Step Functionsrichiede di fornire un ruolo di esecuzione nei seguenti casi:

- Crei una macchina a stati nella console, negli AWS SDK o AWS CLI utilizzando l'CreateStateMachineAPI.
- Puoi testare uno stato nella console, negli AWS SDK o AWS CLI utilizzando l'TestStateAPI.

Workflow Studio dispone di funzionalità che semplificano la gestione dei ruoli di esecuzione per i flussi di lavoro.

Argomenti

- Informazioni sui ruoli generati automaticamente
- Generazione automatica di ruoli
- Risoluzione dei problemi relativi alla generazione di ruoli
- Ruolo per il test delle attività HTTP in Workflow Studio
- Ruolo per testare un'integrazione di servizi ottimizzata in Workflow Studio
- Ruolo per testare l'integrazione di un servizio AWS SDK in Workflow Studio
- Ruolo per testare gli stati del flusso in Workflow Studio

Informazioni sui ruoli generati automaticamente

Quando crei una macchina a stati nella Step Functions console, Workflow Studio può creare automaticamente un ruolo di esecuzione che contiene le IAM politiche necessarie. Workflow Studio analizza la definizione della macchina a stati e genera politiche con i privilegi minimi necessari per eseguire il flusso di lavoro.

Workflow Studio può generare IAM policy per quanto segue:

- Attività HTTP che richiamano API di terze parti.
- Stati di attività che chiamano altri Servizi AWS utenti utilizzando integrazioni ottimizzate, come LambdaInvoke, DynamoDB GetItemo. AWS Glue StartJobRun
- Stati delle attività che eseguono flussi di lavoro annidati.
- Stati della mappa distribuita, incluse le politiche per avviare esecuzioni secondarie di flussi di lavoro, elencare Amazon S3 i bucket e leggere o scrivere oggetti S3.
- X-Raytracciamento. Ogni ruolo generato automaticamente in Workflow Studio contiene una politica che concede le autorizzazioni alla macchina a stati a cui inviare tracce. X-Ray
- Registrazione utilizzandoCloudWatchRegistriquando la registrazione è abilitata sulla macchina a stati.

Workflow Studio non è in grado IAM di generare policy per gli stati delle attività che chiamano altri Servizi AWS utilizzando integrazioni AWS SDK.

Generazione automatica di ruoli

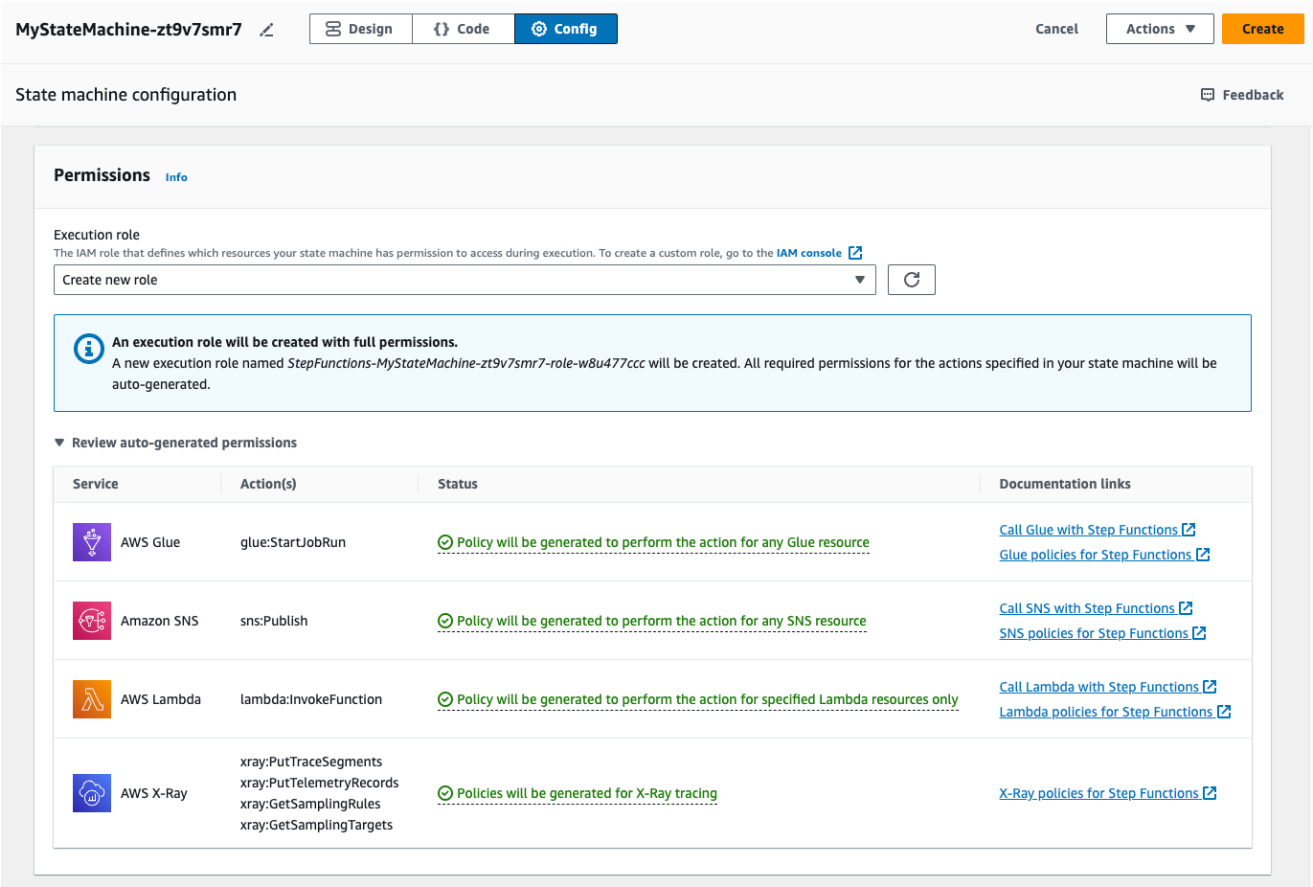
1. Apri la console Step Functions e scegli Crea macchina a stati.

Puoi anche aggiornare una macchina a stati esistente. Fai riferimento al Passaggio 4 se stai aggiornando una macchina a stati.

2. Nella finestra di dialogo Scegli un modello, seleziona Vuoto.
3. Scegli Seleziona. Questo apre Workflow Studio in[Modalità di progettazione](#).
4. Scegli la scheda Config.
5. Scorri verso il basso fino alla sezione Autorizzazioni ed esegui le seguenti operazioni:

   a. Per il ruolo di esecuzione, assicurati di mantenere la selezione predefinita di Crea nuovo ruolo.

   Workflow Studio genera automaticamente tutte le IAM politiche richieste per ogni stato valido nella definizione della macchina a stati. Visualizza un banner con il messaggio, Verrà creato un ruolo di esecuzione con autorizzazioni complete.



> **Tip**
>
> Per esaminare le autorizzazioni generate automaticamente da Workflow Studio per la tua macchina a stati, scegli Rivedi le autorizzazioni generate automaticamente.

> **Note**
>
> Se elimini il ruolo IAM creato da Step Functions, Step Functions non può ricrearlo in un secondo momento. Allo stesso modo, se modifichi il ruolo (ad esempio, rimuovendo Step Functions dai principi nella policy IAM), Step Functions non può ripristinare le impostazioni originali in un secondo momento.

Se Workflow Studio non è in grado di generare tutte le IAM politiche richieste, visualizza un banner con il messaggio Le autorizzazioni per determinate azioni non possono essere generate automaticamente. Verrà creato un IAM ruolo solo con autorizzazioni parziali. Per informazioni su come aggiungere le autorizzazioni mancanti, consulta. Risoluzione dei problemi relativi alla generazione di ruoli

b. Scegli Crea se stai creando una macchina a stati. Altrimenti, scegliere Save (Salva).

c. Scegli Conferma nella finestra di dialogo che appare.

Workflow Studio salva la macchina a stati e crea il nuovo ruolo di esecuzione.

Risoluzione dei problemi relativi alla generazione di ruoli

Workflow Studio non è in grado di generare automaticamente un ruolo di esecuzione con tutte le autorizzazioni richieste nei seguenti casi:

- Ci sono errori nella tua macchina a stati. Assicurati di risolvere tutti gli errori di convalida in Workflow Studio. Inoltre, assicuratevi di correggere eventuali errori sul lato server riscontrati durante il salvataggio.
- La tua macchina a stati contiene attività che utilizzano integrazioni AWS SDK. Workflow Studio non può generare automaticamente IAM le policy in questo caso. Workflow Studio visualizza un banner con il messaggio «Le autorizzazioni per determinate azioni non possono essere generate automaticamente». Verrà creato un IAM ruolo solo con autorizzazioni parziali. Nella tabella Rivedi

le autorizzazioni generate automaticamente, scegli il contenuto in Stato per ulteriori informazioni sulle politiche mancanti nel tuo ruolo di esecuzione. Workflow Studio può ancora generare un ruolo di esecuzione, ma questo ruolo non conterrà IAM politiche per tutte le azioni. Consulta i collegamenti nella sezione Collegamenti alla documentazione per scrivere le tue politiche e aggiungerle al ruolo dopo che è stato generato. Questi collegamenti sono disponibili anche dopo aver salvato la macchina a stati.

Ruolo per il test delle attività HTTP in Workflow Studio

È necessario un ruolo di esecuzione per testare lo stato di un task HTTP. Se non disponi di un ruolo con autorizzazioni sufficienti, utilizza una delle seguenti opzioni per creare un ruolo:

- Generazione automatica di un ruolo con Workflow Studio (consigliato): questa è l'opzione sicura. Chiudi la finestra di dialogo Test state e segui le istruzioni riportate inGenerazione automatica di ruoli. Ciò richiederà prima di creare o aggiornare la macchina a stati, quindi tornare in Workflow Studio per testare lo stato.
- Usa un ruolo con accesso da amministratore: se disponi delle autorizzazioni per creare un ruolo con accesso completo a tutti i servizi e le risorse AWS, puoi utilizzare quel ruolo per testare qualsiasi tipo di stato nel tuo flusso di lavoro. A tale scopo, puoi creare un ruolo di Step Functions servizio e aggiungervi la AdministratorAccess policy nella IAM console https://console.aws.amazon.com/iam/.

Ruolo per testare un'integrazione di servizi ottimizzata in Workflow Studio

È necessario un ruolo di esecuzione per gli stati Task che richiamano integrazioni di servizi ottimizzate. Se non disponi di un ruolo con autorizzazioni sufficienti, utilizza una delle seguenti opzioni per creare un ruolo:

- Utilizza i collegamenti alla documentazione di Workflow Studio per scrivere IAM le tue politiche (consigliato): questa è l'opzione sicura. Chiudete la finestra di dialogo Test state e seguite le istruzioni riportate inGenerazione automatica di ruoli. Ciò richiederà prima di creare o aggiornare la macchina a stati, quindi tornare in Workflow Studio per testare lo stato.
- Usa un ruolo con accesso da amministratore: se disponi delle autorizzazioni per creare un ruolo con accesso completo a tutti i servizi e le risorse AWS, puoi utilizzare quel ruolo per testare qualsiasi tipo di stato nel tuo flusso di lavoro. A tale scopo, puoi creare un ruolo di

Step Functions servizio e aggiungervi la AdministratorAccess policy nella IAM console https://console.aws.amazon.com/iam/.

Ruolo per testare l'integrazione di un servizio AWS SDK in Workflow Studio

È necessario un ruolo di esecuzione per gli stati Task che richiamano le integrazioni AWS SDK. Se non disponi di un ruolo con autorizzazioni sufficienti, utilizza una delle seguenti opzioni per creare un ruolo:

- Utilizza i collegamenti alla documentazione di Workflow Studio per scrivere IAM le tue politiche (consigliato): questa è l'opzione sicura. Chiudete la finestra di dialogo Test state e seguite le istruzioni riportate inGenerazione automatica di ruoli. Ciò richiederà prima di creare o aggiornare la macchina a stati, quindi tornare in Workflow Studio per testare lo stato. Esegui questa operazione:
  1. Chiudete la finestra di dialogo dello stato di test
  2. Scegli la scheda Config per visualizzare la modalità Config.
  3. Scorri verso il basso fino alla sezione Autorizzazioni.
  4. Workflow Studio visualizza un banner con il messaggio «Le autorizzazioni per determinate azioni non possono essere generate automaticamente». Verrà creato un IAM ruolo solo con autorizzazioni parziali. Scegli Rivedi le autorizzazioni generate automaticamente.
  5. La tabella Rivedi le autorizzazioni generate automaticamente visualizza una riga che mostra l'azione corrispondente allo stato dell'attività che desideri testare. Consulta i collegamenti nella sezione Collegamenti alla documentazione per scrivere IAM le tue politiche in un ruolo personalizzato.
- Usa un ruolo con accesso da amministratore: se disponi delle autorizzazioni per creare un ruolo con accesso completo a tutti i servizi e le risorse AWS, puoi utilizzare quel ruolo per testare qualsiasi tipo di stato nel tuo flusso di lavoro. A tale scopo, puoi creare un ruolo di Step Functions servizio e aggiungervi la AdministratorAccess policy nella IAM console https://console.aws.amazon.com/iam/.

Ruolo per testare gli stati del flusso in Workflow Studio

È necessario un ruolo di esecuzione per testare gli stati del flusso in Workflow Studio. Gli stati del flusso sono quegli stati che dirigono il flusso di esecuzioneChoice, ad esempio ParallelEseguire la mappatura,Pass,Attendi,,Succeed, oFail. L'TestStateAPI non funziona con gli stati Map o Parallel. Utilizza una delle seguenti opzioni per creare un ruolo per testare lo stato di un flusso:

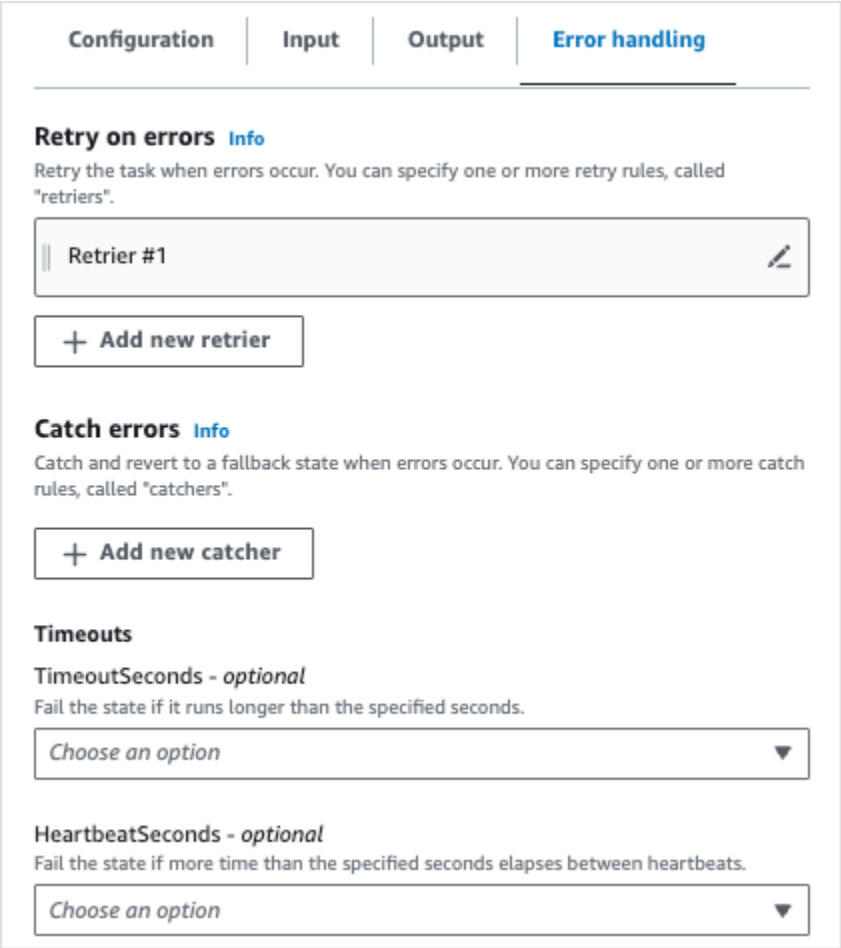
- Usa qualsiasi ruolo nel tuo Account AWS (consigliato): gli stati di flusso non richiedono IAM politiche specifiche, perché non richiedono AWS azioni o risorse. Pertanto, puoi utilizzare qualsiasi IAM ruolo nel tuo Account AWS.

  1. Nella finestra di dialogo Stato del test, selezionate un ruolo dall'elenco a discesa Ruolo di esecuzione.
  2. Se non viene visualizzato alcun ruolo nell'elenco a discesa, procedi come segue:
     a. Nella IAM console https://console.aws.amazon.com/iam/, scegli Ruoli.
     b. Scegliete un ruolo dall'elenco e copiatene l'ARN dalla pagina dei dettagli del ruolo. È necessario fornire questo ARN nella finestra di dialogo Test state.
     c. Nella finestra di dialogo Stato del test, selezionare Inserisci un ruolo ARN dall'elenco a discesa Ruolo di esecuzione.
     d. Incolla l'ARN in Role ARN.

- Usa un ruolo con accesso da amministratore: se disponi delle autorizzazioni per creare un ruolo con accesso completo a tutti i servizi e le risorse AWS, puoi utilizzare quel ruolo per testare qualsiasi tipo di stato nel tuo flusso di lavoro. A tale scopo, puoi creare un ruolo di Step Functions servizio e aggiungervi la AdministratorAccess policy nella IAM console https://console.aws.amazon.com/iam/.

Gestione degli errori

Per impostazione predefinita, quando uno stato segnala un errore, Step Functions fa sì che l'esecuzione del flusso di lavoro fallisca completamente. Per le azioni e alcuni stati del flusso, puoi configurare il modo in cui Step Functions gestisce gli errori. Anche se hai configurato la gestione degli errori, alcuni errori possono comunque causare il fallimento dell'esecuzione del flusso di lavoro. Per ulteriori informazioni, consulta Gestione degli errori in Step Functions. In Workflow Studio, configurate la gestione degli errori nella scheda Gestione degli errori del ✓Inspector pannello.



Riprova sugli errori

È possibile aggiungere una o più regole agli stati di azione e allo stato del Parallel flusso per riprovare l'attività quando si verifica un errore. Queste regole sono chiamate retrier. Per aggiungere un retrier, scegli l'icona di modifica nella casella Retrier #1, quindi configura le sue opzioni:

- (Facoltativo) Nel campo Commento, aggiungi il tuo commento. Non influirà sul flusso di lavoro, ma può essere utilizzato per annotare il flusso di lavoro.
- Posizionate il cursore nel campo Errori e scegliete un errore che attiverà il retrier oppure inserite un nome di errore personalizzato. Puoi scegliere o aggiungere più errori.
- (Facoltativo) Imposta un intervallo. Questo è il tempo in secondi prima che Step Functions effettui il primo tentativo. Seguiranno altri tentativi a intervalli configurabili con Max tentations e Backoff rate.
- (Facoltativo) Imposta il numero massimo di tentativi. Questo è il numero massimo di tentativi prima che Step Functions faccia fallire l'esecuzione.

- (Facoltativo) Imposta la frequenza di backoff. Si tratta di un moltiplicatore che determina di quanto aumenterà l'intervallo tra i tentativi ad ogni tentativo.

> **Note**
>
> Non tutte le opzioni di gestione degli errori sono disponibili per tutti gli stati. Lambda Invoke ha un retrier configurato per impostazione predefinita.

Rileva gli errori

È possibile aggiungere una o più regole agli stati di azione Parallel e Eseguire la mappatura flow per catturare un errore. Queste regole sono chiamate catcher. Per aggiungere un catcher, scegli Aggiungi nuovo catcher, quindi configura le sue opzioni:

- (Facoltativo) Nel campo Commento, aggiungi il tuo commento. Non influirà sul flusso di lavoro, ma può essere utilizzato per annotare il flusso di lavoro.
- Posiziona il cursore nel campo Errori e scegli un errore che attiverà il rilevatore oppure inserisci un nome di errore personalizzato. Puoi scegliere o aggiungere più errori.
- Nel campo Fallback state, scegli uno stato di fallback. Questo è lo stato successivo in cui passerà il flusso di lavoro dopo che viene rilevato un errore.
- (Facoltativo) Nel ResultPathcampo, aggiungete un `ResultPath` filtro per aggiungere l'errore all'input di stato originale. `ResultPath`Deve essere valido JsonPath. Questo verrà inviato allo stato di fallback.

Timeout

È possibile configurare un timeout per gli stati di azione per impostare il numero massimo di secondi di esecuzione dello stato prima che si verifichi un errore. Utilizza i timeout per evitare esecuzioni bloccate. Per configurare un timeout, inserisci il numero di secondi che lo stato deve attendere prima che l'esecuzione abbia esito negativo. Per ulteriori informazioni sui timeout, vedi `TimeoutSeconds` in state. Attività

HeartbeatSeconds

Puoi configurare un Heartbeat o una notifica periodica inviata dall'attività. Se imposti un intervallo di battito cardiaco e il tuo stato non invia notifiche relative al battito cardiaco negli intervalli

configurati, l'operazione viene contrassegnata come non riuscita. Per configurare un battito cardiaco, impostate un numero intero positivo diverso da zero di secondi. Per ulteriori informazioni, vedere `HeartBeatSeconds` in state. Attività

Tutorial: Impara a usare AWS Step Functions Workflow Studio

In questo tutorial, imparerai le basi per lavorare con Workflow Studio per AWS Step Functions. In Modalità di progettazione Workflow Studio, creerai una macchina a stati contenente più stati `PassChoice`, `FailWait`, e`Parallel`. Utilizzerai la funzione drag and drop per cercare, selezionare e configurare questi stati. Quindi, visualizzerai la definizione generata automaticamente Amazon States Language (ASL) del tuo flusso di lavoro. Utilizzerai anche Workflow Studio per modificare la Modalità codice definizione del flusso di lavoro. Quindi, uscirete da Workflow Studio, avvierete la macchina a stati ed esaminerete i dettagli di esecuzione.

In questo tutorial, imparerete anche come aggiornare la macchina a stati e visualizzare le modifiche nell'output di esecuzione. Infine, eseguirete una fase di pulizia ed eliminerete la vostra macchina a stati.

Dopo aver completato questo tutorial, saprete come utilizzare Workflow Studio per creare e configurare un flusso di lavoro utilizzando le modalità Progettazione e Codice. Saprete anche come aggiornare, eseguire ed eliminare la vostra macchina a stati.

> **Note**
>
> Prima di iniziare, assicurati di completare i prerequisiti per questo tutorial.

Argomenti

- Passaggio 1: accedi a Workflow Studio
- Fase 2: Creare una macchina a stati
- Fase 3: Rivedi la definizione di Amazon States Language generata automaticamente
- Fase 4: Modificare la definizione del flusso di lavoro in modalità Codice
- Fase 5: Salvare la macchina a stati
- Passaggio 6: Eseguire la macchina a stati
- Passaggio 7: aggiorna la tua macchina a stati

- Fase 8: Pulizia

Passaggio 1: accedi a Workflow Studio

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Nella finestra di dialogo Scegli un modello, seleziona Vuoto.
3. Scegli Seleziona. Questo apre Workflow Studio inModalità di progettazione.

Fase 2: Creare una macchina a stati

In Workflow Studio, una macchina a stati è una rappresentazione grafica del flusso di lavoro. Con Workflow Studio, è possibile definire, configurare ed esaminare le singole fasi del flusso di lavoro. Nei passaggi seguenti, si utilizza Workflow Studio per creare la propria macchina a stati. Modalità di progettazione

Per creare una macchina a stati

1. Assicurati di essere nella modalità Progettazione di Workflow Studio.
2. Da Stati del browser sinistra, scegli la scheda Flusso. Quindi, trascina uno stato Pass sullo stato vuoto denominato Trascina il primo stato qui.
3. Trascina uno stato Choice dalla scheda Flow e rilascialo sotto lo stato Pass.
4. Per State name, sostituisci il nome predefinito di Choice. Per questo tutorial, utilizza il nome **`IsHelloWorldExample`**.
5. Trascina un altro stato Pass e rilascialo su un ramo dello IsHelloWorldExamplestato. Quindi, trascina uno stato Fail e rilascialo sotto l'altro ramo dello IsHelloWorldExamplestato.
6. Scegli lo stato Pass (1) e rinominalo in**`Yes`**. Rinomina lo stato Fail come. **`No`**
7. Specificate la logica di ramificazione IsHelloWorldExampledello stato utilizzando la variabile booleana. `IsHelloWorldExample`

   In caso **`IsHelloWorldExample`** affermativo**`False`**, il flusso di lavoro passerà allo stato No. In caso contrario, il flusso di lavoro continuerà il flusso di esecuzione nello stato Sì.

   Per definire la logica di ramificazione, completa le operazioni seguenti:

   a. Scegliete lo IsHelloWorldExamplestato sulloCanvas, quindi in Regole di scelta, scegliete l'icona di modifica nella casella Regola #1 per definire la regola di prima scelta.

b. Scegli Aggiungi condizione.

c. Nella finestra di dialogo Condizioni per la regola #1, immettete **`$.IsHelloWorldExample`** in Variabile.

d. Scegli è uguale a Operatore.

e. Scegli costante booleana in Valore, quindi scegli true dall'elenco a discesa.

f. Scegli Salva condizioni.

g. Assicurati che nel menu a discesa Poi next state sia: l'opzione Sì sia selezionata.

h. Scegli Aggiungi nuova regola di scelta, quindi scegli Aggiungi condizioni.

i. Nella casella Regola #2, definisci la regola della seconda scelta quando il valore della `IsHelloWorldExample` variabile è falso ripetendo i passaggi secondari da 7.c a 7.f. Per il passaggio 7.e, scegli false anziché true.

j. Nella casella Regola #2, scegli No dall'elenco a discesa Allora lo stato successivo è:.

k. Nella casella Regola predefinita, scegli l'icona di modifica per definire la regola di scelta predefinita, quindi scegli Sì dall'elenco a discesa.

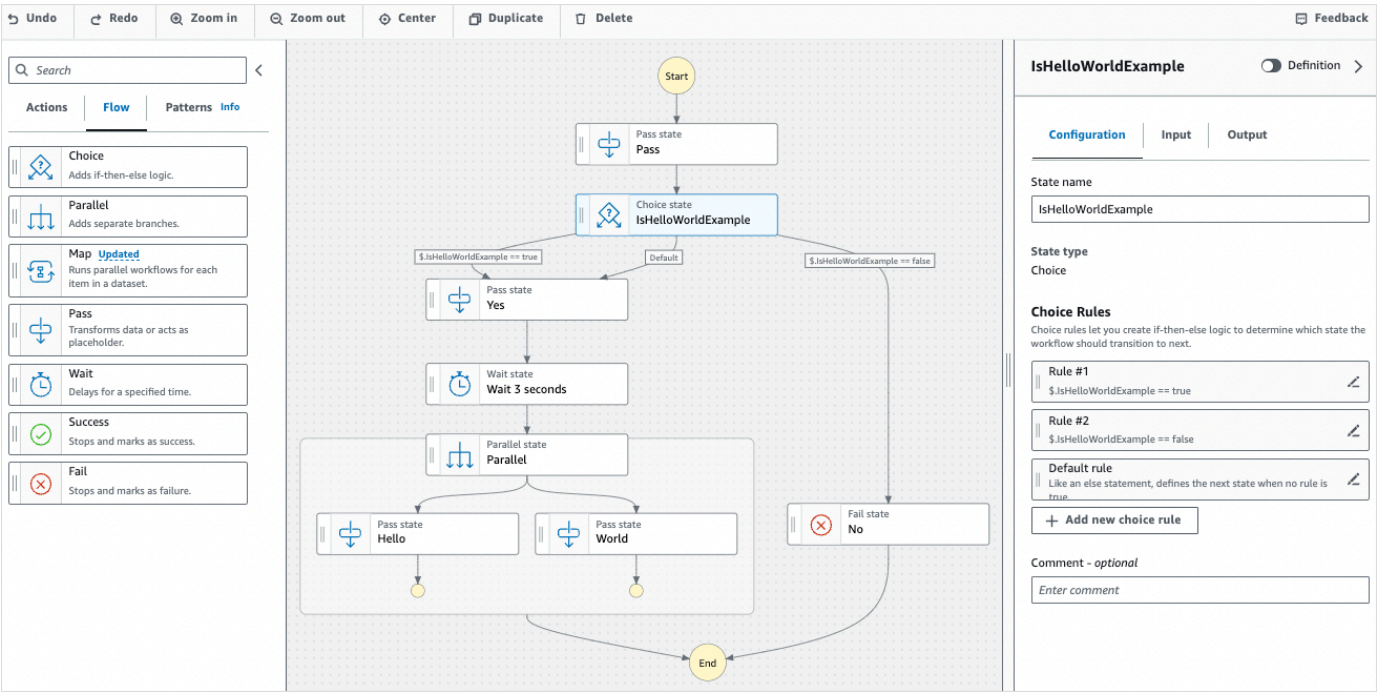
8. Aggiungi uno stato di attesa dopo lo stato Sì e assegnagli un **`Wait 3 sec`** nome. Quindi, configura il tempo di attesa in modo che sia di tre secondi effettuando le seguenti operazioni:

   a. In Opzioni, mantieni la selezione predefinita di Attendi un intervallo fisso.

   b. In Secondi, assicurati che sia selezionato Inserisci secondi, quindi inserisci **`3`** nella casella.

9. Dopo lo stato Wait 3 sec, aggiungi uno stato Parallel. Aggiungi due stati Pass nei suoi due rami. Assegna un nome al primo stato Pass**`Hello`**. Assegna un nome al secondo stato del Pass**`World`**.

   Il flusso di lavoro completato avrà il seguente aspetto:



Fase 3: Rivedi la definizione di Amazon States Language generata automaticamente

Mentre trascini gli stati dalla scheda Flow sull'area di disegno, Workflow Studio compone automaticamente la definizione Amazon States Language (ASL) del tuo flusso di lavoro in tempo reale. Nel ✓Inspector pannello, scegli l'interruttore Definizione per visualizzare questa definizione o passa al pulsante per modificare la definizione Modalità codice secondo necessità. Per informazioni sulla modifica della definizione del flusso di lavoro, consultate il passaggio 4 di questo tutorial.

- (Facoltativo) Scegliete Definizione nel pannello Inspector e visualizzate il flusso di lavoro della macchina a stati.

  Il codice di esempio seguente mostra la definizione di Amazon States Language generata automaticamente per la macchina a `IsHelloWorldExample` stati. Lo `Choice` stato aggiunto in Workflow Studio viene utilizzato per determinare il flusso di esecuzione in base alla logica di ramificazione definita nella Fase 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
            }
          }
        }
      ]
    }
  }
}
```

Fase 4: Modificare la definizione del flusso di lavoro in modalità Codice

La modalità Codice di Workflow Studio fornisce un editor di codice integrato per visualizzare e modificare la definizione ASL dei flussi di lavoro.

1. Scegliete Code per passare alla modalità Codice.
2. Dopo la definizione dello stato parallelo, posiziona il cursore e premi**`Enter`**.
3. Premete **`Ctrl+space`** per visualizzare l'elenco degli stati che potete aggiungere dopo lo stato Parallel.
4. Scegli Pass State dall'elenco di opzioni. L'editor di codice aggiunge codice standard per Pass State.
5. L'aggiunta di questo stato comporta errori nella definizione del flusso di lavoro. Nella definizione dello stato parallelo, sostituisci `"End": true` con**`"Next": "PassState"`**.
6. Nella definizione di Pass State che hai aggiunto, apporta le seguenti modifiche:
   a. Rimuovi il nodo Risultato.
   b. Rimuovi `"ResultPath": "$.result"`, e`"Next": "NextState"`.

c. Dopo`"Type": "Pass",`, entra**`"End": true`**.

d. Aggiungi un **`,`** dopo la definizione del Pass State.

La definizione del flusso di lavoro dovrebbe ora essere simile alla definizione seguente.

```
{
  "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"
    }
  }
}
```

Fase 5: Salvare la macchina a stati

1. Scegli l'altro Config o scegli l'icona di modifica accanto al nome della macchina a stati predefinita di. MyStateMachine Nella configurazione della macchina a stati, specifica un nome. Ad esempio, specifica **`HelloWorld`**.
2. (Facoltativo) Specificate altre impostazioni del flusso di lavoro, come il tipo di macchina a stati e il relativo ruolo di esecuzione. Per questo tutorial, mantieni tutte le selezioni predefinite nella configurazione della macchina a stati.
3. Scegli Crea.
4. Nella finestra di dialogo Conferma creazione del ruolo, scegli Conferma per continuare.

Puoi anche scegliere Visualizza la configurazione del ruolo per tornare alla modalità Config.

Per ulteriori informazioni sulla modalità Config, vedere Modalità Config di Workflow Studio.

Passaggio 6: Eseguire la macchina a stati

Le esecuzioni di macchine a stati sono casi in cui si esegue il flusso di lavoro per eseguire attività.

1. Nella pagina Macchine a stati, scegli la macchina a HelloWorldstati.
2. Nella HelloWorldpagina, scegli Avvia esecuzione.
3. (Facoltativo) Per identificare l'esecuzione, puoi specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

   > **Note**
   >
   > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

4. Nella casella Input, inserisci i valori di input per l'esecuzione in formato JSON. In base all'input, la `IsHelloWorldExample` variabile determina quale flusso della macchina a stati verrà eseguito. Per ora, usa il seguente valore di input:

   ```
   {
      "IsHelloWorldExample": true
   }
   ```

   > **Note**
   >
   > Sebbene specificare un input di esecuzione sia facoltativo, in questo tutorial è obbligatorio specificare un input di esecuzione simile all'input di esempio precedente. Questo valore di input viene referenziato nello `Choice` stato quando si esegue la macchina a stati.

5. Selezionare Start execution (Avvia esecuzione).

6. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

   Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

   Per questo tutorial, se hai inserito un valore di input di`"IsHelloWorldExample": true`, dovresti vedere il seguente risultato:

   ```
   {
      "IsHelloWorldExample": true
   },
   {
      "IsHelloWorldExample": true
   }
   ```

Passaggio 7: aggiorna la tua macchina a stati

Quando aggiorni una macchina a stati, gli aggiornamenti alla fine sono coerenti. Dopo un breve periodo di tempo, tutte le esecuzioni appena avviate rifletteranno la definizione aggiornata della macchina a stati. Tutte le esecuzioni attualmente in esecuzione verranno eseguite fino al completamento in base alla definizione precedente.

In questo passaggio, aggiornerai la tua macchina a stati nella Modalità di progettazione modalità Workflow Studio. Aggiungerai un `Result` campo nello stato del Pass denominato World.

1. Nella pagina intitolata con il tuo ID di esecuzione, scegli Modifica macchina a stati.
2. Assicurati di essere in modalità Progettazione.
3. Scegli lo stato del Pass denominato World sull'area di disegno, quindi scegli Output.
4. Nella casella Risultato, immettete**`"World has been updated!"`**.
5. Selezionare Salva.
6. (Facoltativo) Nell'area Definizione, visualizza la definizione aggiornata di Amazon States Language del tuo flusso di lavoro.

```
{
      "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. Scegliere Execute (Esegui).
8. Nella finestra di dialogo Avvia esecuzione che si apre in una nuova scheda, fornisci il seguente input di esecuzione.

```
{
   "IsHelloWorldExample": true
}
```

9. Selezionare Start Execution (Avvia esecuzione).
10. (Facoltativo) Nella vista Grafico, scegliete il passo World, quindi scegliete Output. L'output è World has been updated!

Fase 8: Pulizia

Per eliminare la tua macchina a stati

1. Dal menu di navigazione, scegli Macchine a stati.
2. Nella pagina Macchine a stati, selezionate HelloWorld, quindi scegliete Elimina.
3. Nella finestra di dialogo Elimina macchina a stati, digitate **`delete`** per confermare l'eliminazione.
4. Scegli Elimina.

   Se l'eliminazione ha esito positivo, nella parte superiore dello schermo viene visualizzata una barra di stato verde. La barra di stato verde indica che la macchina a stati è contrassegnata per l'eliminazione. La tua macchina a stati verrà eliminata quando tutte le sue esecuzioni in corso smetteranno di funzionare.

Per eliminare il tuo ruolo di esecuzione

1. Apri la pagina Ruoli per IAM.
2. Scegli il ruolo IAM che Step Functions ha creato per te. Ad esempio, StepFunctions- HelloWorld - Role-example.
3. Scegliere Delete role (Elimina ruolo).
4. Scegliere Yes, delete (Sì, elimina).

Tutorial per Step Functions

I tutorial in questa sezione ti aiutano a comprendere diversi aspetti dell'utilizzo di AWS Step Functions.

Per completare questi tutorial, è necessario un account. AWS Se non hai un AWS account, vai su https://aws.amazon.com/ e scegli Crea un AWS account.

Argomenti

- Creazione di una macchina a stati Step Functions che utilizza Lambda
- Gestione delle condizioni di errore utilizzando una macchina a stati Step Functions
- Utilizzo dello stato della mappa in linea per ripetere un'azione
- Copiare dati CSV su larga scala utilizzando Distributed Map
- Elaborazione di un intero batch di dati con una funzione Lambda
- Elaborazione di singoli elementi di dati con una funzione Lambda
- Avvio dell'esecuzione di una macchina a stati in risposta agli eventi di Amazon S3
- Creazione di un'API Step Functions utilizzando API Gateway
- Creazione di una macchina a stati Step FunctionsAWS SAM
- Creazione di una macchina a stati Activity utilizzando Step Functions
- Iterare un ciclo con Lambda
- Esecuzioni continue di flussi di lavoro a lungo termine come nuova esecuzione
- Distribuire un progetto di approvazione umana di esempio
- Visualizza le tracce X-Ray in Step Functions
- Raccogli informazioni sui bucket Amazon S3 utilizzando AWS le integrazioni dei servizi SDK

Creazione di una macchina a stati Step Functions che utilizza Lambda

In questo tutorial, creerai un flusso di lavoro in un unico passaggio utilizzando AWS Step Functions per richiamare una AWS Lambda funzione.

> **Note**
>
> Step Functions si fonda su attività e macchine a stati. In Step Functions, un flusso di lavoro viene chiamato macchina a stati, che consiste in una serie di passaggi basati sugli eventi. Ogni fase di un flusso di lavoro è denominata stato. Uno Attività stato rappresenta un'unità di lavoro eseguita da un altro AWS servizio, ad esempio AWS Lambda. Uno stato Task può chiamare Servizio AWS qualsiasi API.
> Per ulteriori informazioni, consultare:
>
> - Che cos'è AWS Step Functions?
> - Chiama altri servizi AWS

Lambda è ideale per `Task` gli stati, perché le funzioni Lambda sono senza server e facili da scrivere. Puoi scrivere codice nel AWS Management Console o nel tuo editor preferito. AWS gestisce i dettagli relativi alla fornitura di un ambiente informatico per la funzione e alla sua esecuzione.

In questo argomento:

- Fase 1: Creazione di una funzione Lambda
- Fase 2: testare la funzione Lambda
- Passaggio 3: Creare una macchina a stati
- Fase 4: Esegui la macchina a stati

Fase 1: Creazione di una funzione Lambda

La funzione Lambda riceve i dati degli eventi e restituisce un messaggio di saluto.

> **Important**
>
> Assicurati che la tua funzione Lambda utilizzi lo stesso AWS account e la stessa AWS regione della tua macchina a stati.

1. Apri la console Lambda e scegli Crea funzione.
2. Nella pagina Create function (Crea funzione), scegliere Author from scratch (Crea da zero).
3. Nel campo Function name (Nome funzione), immettere `HelloFunction`.

4. Mantieni le selezioni predefinite per tutte le altre opzioni, quindi scegli Crea funzione.
5. Dopo aver creato la funzione Lambda, copia l'Amazon Resource Name (ARN) della funzione visualizzato nell'angolo in alto a destra della pagina. Per copiare l'ARN, fate clic su.

   Di seguito è riportato un esempio di ARN:

   ```
   arn:aws:lambda:us-east-1:123456789012:function:HelloFunction
   ```

6. Copia il codice seguente per la funzione Lambda nella sezione Codice sorgente della ***HelloFunction***pagina.

   ```
   export const handler = async(event, context, callback) => {
       callback(null, "Hello from " + event.who + "!");
   };
   ```

   Questo codice assembla un saluto utilizzando il campo `who` dei dati di input, forniti dall'oggetto `event` passato alla funzione. I dati di input per questa funzione vengono aggiunti in seguito, quando si avvia una nuova esecuzione. Il metodo `callback` restituisce il saluto assemblato dalla funzione.
7. Seleziona Deploy (Implementa).

Fase 2: testare la funzione Lambda

Prova la tua funzione Lambda per vederla in funzione.

1. Scegli Test (Esegui test).
2. Per Event name (Nome evento) immettere `HelloEvent`.
3. Sostituisci i dati JSON degli eventi con i seguenti.

   ```
   {
       "who": "AWS Step Functions"
   }
   ```

   La `"who"` voce corrisponde al `event.who` campo della funzione Lambda, che completa il saluto. Immetterai gli stessi dati di input quando eseguirai la tua macchina a stati.
4. Scegli Salva, quindi scegli Test.

5. Per esaminare i risultati del test, in Execution result (Risultato esecuzione), espandi Details (Dettagli).

Passaggio 3: Creare una macchina a stati

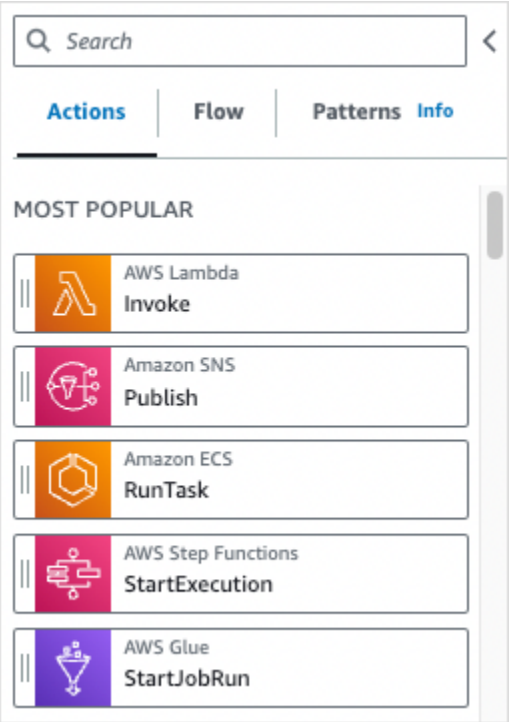
Usa la console Step Functions per creare una macchina a stati che richiami la funzione Lambda creata nel passaggio 1.

1. Apri la console Step Functions e scegli Crea macchina a stati.

   > **Important**
   >
   > Assicurati che la tua macchina a stati utilizzi lo stesso AWS account e la stessa regione della funzione Lambda che hai creato in precedenza.

2. Nella finestra di dialogo Scegli un modello, seleziona Vuoto.
3. Scegli Seleziona. Questo apre Workflow Studio inModalità di progettazione.
4. Nel browser States a sinistra, assicurati di aver scelto la scheda Azioni. Successivamente, esegui queste operazioni:

   - Trascina e rilascia l'API AWS Lambda Invoke nello stato vuoto denominato Drag first state here.



5. Nel pannello Inspector a destra, configura la funzione Lambda:

   a. Nella sezione Parametri API, scegli la funzione Lambda che hai creato in precedenza nell'elenco a discesa Nome funzione.

   b. Mantieni la selezione predefinita nell'elenco a discesa Payload.

6. (Facoltativo) Scegliete Definizione per visualizzare la definizione della macchina a stati Amazon States Language (ASL), che viene generata automaticamente in base alle selezioni effettuate nella scheda Azioni e nel pannello Inspector.

7. Specificate un nome per la vostra macchina a stati. Per fare ciò, scegli l'icona di modifica accanto al nome della macchina a stati predefinita di MyStateMachine. Quindi, nella configurazione della macchina a stati, specifica un nome nella casella Nome macchina a stati.

   Ad esempio, è possibile inserire il nome **`LambdaStateMachine`**.

   > **Note**
   >
   > I nomi delle macchine a stati, delle esecuzioni e delle attività non devono superare gli 80 caratteri di lunghezza. Questi nomi devono essere univoci per l'account e la AWS regione e non devono contenere nessuno dei seguenti elementi:
   >
   > - Spazi bianchi

- Caratteri jolly () `?` `*`
- Caratteri tra parentesi quadre () `<` `>` `{` `}` `[` `]`
- Caratteri speciali () `"` `#` `%` `\` `^` `|` `~` `` ` `` `$` `&` `,` `;` `:` `/`
- Caratteri di controllo (`\\u0000`- `\\u001f` o `\\u007f` -`\\u009f`).

Se la macchina a stati è di tipo Express, è possibile fornire lo stesso nome a più esecuzioni della macchina a stati. Step Functions genera un ARN di esecuzione univoco per ogni esecuzione della macchina a stati Express, anche se più esecuzioni hanno lo stesso nome.
Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

8. (Facoltativo) Nella configurazione della macchina a stati, specificate altre impostazioni del flusso di lavoro, come il tipo di macchina a stati e il relativo ruolo di esecuzione.

   Per questo tutorial, mantieni tutte le selezioni predefinite nelle impostazioni della macchina a stati.

9. Scegli Crea.
10. Nella finestra di dialogo Conferma creazione del ruolo, scegliete Conferma per continuare.

    Puoi anche scegliere Visualizza le impostazioni del ruolo per tornare alla configurazione della macchina a stati.

> **Note**
>
> Se elimini il ruolo IAM creato da Step Functions, Step Functions non può ricrearlo in un secondo momento. Allo stesso modo, se modifichi il ruolo (ad esempio, rimuovendo Step Functions dai principi nella policy IAM), Step Functions non può ripristinare le impostazioni originali in un secondo momento.

Fase 4: Esegui la macchina a stati

Dopo aver creato la macchina a stati, puoi eseguirla.

1. Nella pagina Macchine a stati, scegli LambdaStateMachine.
2. Selezionare Start execution (Avvia esecuzione).

   Viene visualizzata la finestra di dialogo Avvia esecuzione.
3. (Facoltativo) Per identificare l'esecuzione, è possibile specificare un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

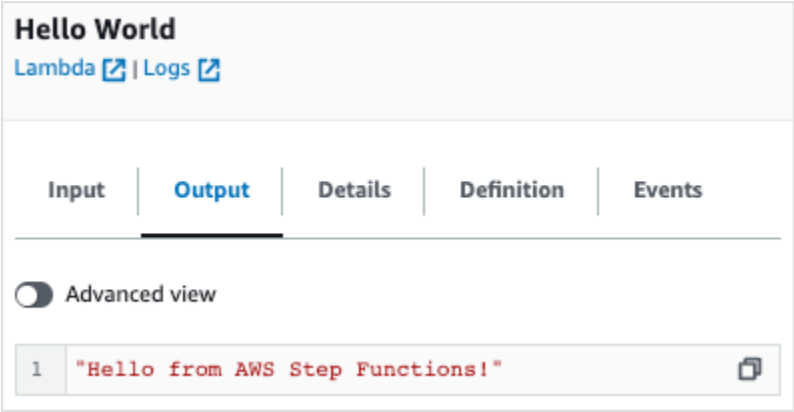
   > **Note**
   >
   > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.
4. Nell'area Input, sostituite i dati di esecuzione di esempio con i seguenti.

   ```
   {
       "who" : "AWS Step Functions"
   }
   ```

   `"who"`è il nome chiave utilizzato dalla funzione Lambda per ottenere il nome della persona da salutare.
5. Selezionare Start Execution (Avvia esecuzione).

   L'esecuzione della macchina a stati inizia e viene visualizzata una nuova pagina che mostra l'esecuzione in esecuzione.
6. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

   Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vedere Pagina dei dettagli di esecuzione: panoramica dell'interfaccia.



> **Note**
>
> Puoi anche passare payload mentre richiami Lambda da una macchina a stati. Per ulteriori informazioni ed esempi su come richiamare Lambda inserendo il payload nel `Parameters` campo, vedere. Richiama Lambda con Step Functions

Gestione delle condizioni di errore utilizzando una macchina a stati Step Functions

In questo tutorial, crei una macchina a AWS Step Functions stati con un Stati di fallback campo. Il `Catch` campo utilizza una AWS Lambda funzione per rispondere con una logica condizionale basata sul tipo di messaggio di errore. Si tratta di una tecnica chiamata gestione degli errori di funzione.

Per ulteriori informazioni, consulta gli errori di AWS Lambda funzione in Node.js nella Guida per gli AWS Lambda sviluppatori.

> **Note**
>
> È inoltre possibile creare macchine a stati che riprovano in base ai timeout o quelle utilizzate `Catch` per passare a uno stato specifico quando si verifica un errore o un timeout. Per esempi di queste tecniche di gestione degli errori, consultare Esempi di utilizzo di Retry e di Catch.

In questo argomento:

- Fase 1: Creare una funzione Lambda che fallisce
- Fase 2: testare la funzione Lambda
- Fase 3: Creare una macchina a stati con un campo Catch
- Fase 4: Esegui la macchina a stati

Fase 1: Creare una funzione Lambda che fallisce

Usa una funzione Lambda per simulare una condizione di errore.

> **Important**
>
> Assicurati che la tua funzione Lambda utilizzi lo stesso AWS account e la stessa AWS regione della tua macchina a stati.

1. Apri la AWS Lambda console all'indirizzo https://console.aws.amazon.com/lambda/.
2. Scegli Crea funzione.
3. Scegli Usa un progetto, `step-functions` entra nella casella di ricerca, quindi scegli il progetto Genera un progetto di errore personalizzato.
4. Nel campo Function name (Nome funzione), immettere `FailFunction`.
5. Per Ruolo, mantieni la selezione predefinita (Crea un nuovo ruolo con autorizzazioni Lambda di base).
6. Il codice seguente viene visualizzato nel riquadro del codice della funzione 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!');
};
```

   L'oggetto `context` restituisce il messaggio di errore `This is a custom error!`.
7. Scegli Crea funzione.

8. Dopo aver creato la funzione Lambda, copia l'Amazon Resource Name (ARN) della funzione visualizzato nell'angolo superiore destro della pagina. Per copiare l'ARN, fate clic su.

   Di seguito è riportato un esempio di ARN:

   ```
   arn:aws:lambda:us-east-1:123456789012:function:FailFunction
   ```

9. Seleziona Deploy (Implementa).

Fase 2: testare la funzione Lambda

Prova la tua funzione Lambda per vederla in funzione.

1. Nella FailFunctionpagina, scegli la scheda Test, quindi scegli Test. Non è necessario creare un evento di test.
2. Per esaminare i risultati del test (l'errore simulato), in Risultato dell'esecuzione, espandi Dettagli.

Fase 3: Creare una macchina a stati con un campo Catch

Usa la console Step Functions per creare una macchina a stati che utilizza uno Attività stato con un `Catch` campo. Aggiungi un riferimento alla tua funzione Lambda nello stato Task. La macchina a stati richiama la funzione Lambda, che fallisce durante l'esecuzione. Step Functions riprova la funzione due volte utilizzando un backoff esponenziale tra un tentativo e l'altro.

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Nella finestra di dialogo Scegli un modello, seleziona Vuoto.
3. Scegli Seleziona. Questo apre Workflow Studio inModalità di progettazione.
4. Scegli Codice per aprire l'editor di codice. Nell'editor di codice, scrivi e modifichi la definizione Amazon States Language (ASL) dei tuoi flussi di lavoro.
5. Incolla il codice seguente, ma sostituisci l'ARN della funzione Lambda che hai creato in precedenza nel campo. `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
      }
   }
}
```

Questa è una descrizione della tua macchina a stati che utilizza Amazon States Language. Definisce un singolo stato `Task` denominato `CreateAccount`. Per ulteriori informazioni, consultare Struttura di una macchina a stati.

Per ulteriori informazioni sulla sintassi del campo `Retry`, consulta Esempi di macchine a stati che utilizzano Retry e Catch.

> **Note**
>
> Gli errori non gestiti in Lambda vengono riportati come `Lambda.Unknown` nell'output degli errori. Questi includono out-of-memory errori e timeout delle funzioni. Puoi abbinare o `States.TaskFailed` gestire questi errori. `Lambda.Unknown` `States.ALL` Quando Lambda raggiunge il numero massimo di chiamate, l'errore è. `Lambda.TooManyRequestsException` Per ulteriori informazioni sugli errori della funzione Lambda, consulta Gestione degli errori e tentativi automatici nella Developer Guide.AWS Lambda

6. (Facoltativo) InRiquadro di visualizzazione del grafico, guarda la visualizzazione grafica in tempo reale del flusso di lavoro.
7. Specificate un nome per la vostra macchina a stati. Per fare ciò, scegli l'icona di modifica accanto al nome della macchina a stati predefinita di MyStateMachine. Quindi, nella configurazione della macchina a stati, specifica un nome nella casella Nome macchina a stati.

   Per questo tutorial, digita **`Catchfailure`**.
8. (Facoltativo) Nella configurazione della macchina a stati, specificate altre impostazioni del flusso di lavoro, come il tipo di macchina a stati e il relativo ruolo di esecuzione.

   Per questo tutorial, mantieni tutte le selezioni predefinite nelle impostazioni della macchina a stati.
9. Nella finestra di dialogo Conferma creazione del ruolo, scegliete Conferma per continuare.

   Puoi anche scegliere Visualizza le impostazioni del ruolo per tornare alla configurazione della macchina a stati.

> **Note**
>
> Se elimini il ruolo IAM creato da Step Functions, Step Functions non può ricrearlo in un secondo momento. Allo stesso modo, se modifichi il ruolo (ad esempio, rimuovendo Step Functions dai principi nella policy IAM), Step Functions non può ripristinare le impostazioni originali in un secondo momento.

Fase 4: Esegui la macchina a stati

Dopo aver creato la macchina a stati, puoi eseguirla.

1. Nella pagina Macchine a stati, scegli Catchfailure.
2. Nella pagina Catchfailure, scegli Avvia esecuzione. Viene visualizzata la finestra di dialogo Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

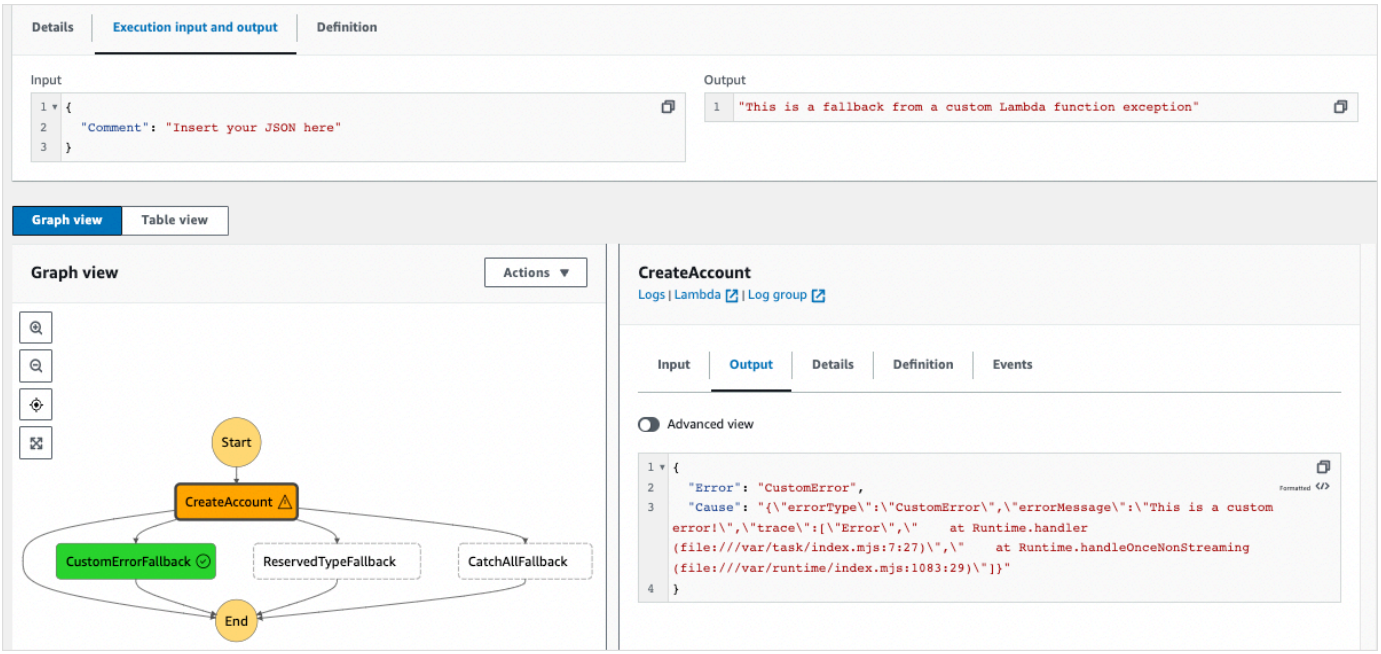
   1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

      > **Note**
      >
      > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

   2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.
   3. Selezionare Start execution (Avvia esecuzione).
   4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

      Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vedere Pagina dei dettagli di esecuzione: panoramica dell'interfaccia.

   Ad esempio, per visualizzare il messaggio di errore personalizzato, scegliete il CreateAccountpassaggio nella visualizzazione Grafico, quindi scegliete la scheda Output.



> **Note**
>
> Puoi conservare l'input di stato insieme all'errore utilizzando `ResultPath`. Per informazioni, consulta [ResultPath Da utilizzare per includere sia l'errore che l'input in un `Catch`](#).

Utilizzo dello stato della mappa in linea per ripetere un'azione

Questo tutorial ti aiuta a iniziare a usare lo `Map` stato in modalità Inline. Lo stato Inline Map viene utilizzato nei flussi di lavoro per eseguire ripetutamente un'azione. Per ulteriori informazioni sulla modalità Inline, consulta Map state in Inline mode.

In questo tutorial, utilizzi lo stato Inline Map per generare ripetutamente identificatori univoci universali della versione 4 (UUID v4). Si inizia creando un flusso di lavoro che contiene due Pass stati e uno stato della mappa in linea in Workflow Studio. Quindi, configurate l'input e l'output, incluso l'array JSON di input per lo `Map` stato. Lo `Map` stato restituisce un array di output che contiene gli UUID v4 generati per ogni elemento dell'array di input.

Indice

- Fase 1: Creare il prototipo del flusso di lavoro
- Fase 2: Configurare input e output

- Fase 3: rivedi la definizione di Amazon States Language generata automaticamente e salva il flusso di lavoro
- Fase 4: Esegui la macchina a stati

Fase 1: Creare il prototipo del flusso di lavoro

In questo passaggio, crei il prototipo per il tuo flusso di lavoro utilizzando Workflow Studio. Workflow Studio è un designer visivo di flussi di lavoro disponibile nella console Step Functions. Sceglierai gli stati richiesti dalla scheda Flow e utilizzerai la funzionalità drag and drop di Workflow Studio per creare il prototipo del flusso di lavoro.

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Nella finestra di dialogo Scegli un modello, seleziona Vuoto.
3. Scegli Seleziona. Questo apre Workflow Studio inModalità di progettazione.
4. Dalla scheda Flow, trascina uno stato Pass e rilascialo nello stato vuoto denominato Drag first state here.
5. Trascina uno stato della mappa e rilascialo sotto lo stato del Pass. Rinomina lo stato della mappa in. **`Map demo`**
6. Trascina un secondo stato Pass e rilascialo all'interno dello stato demo della Mappa.
7. Rinomina il secondo stato Pass in. **`Generate UUID`**

Fase 2: Configurare input e output

In questo passaggio, configuri input e output per tutti gli stati del prototipo del flusso di lavoro. Innanzitutto, inserite alcuni dati fissi nel flusso di lavoro utilizzando il primo stato Pass. Questo stato Pass trasmette questi dati come input allo stato demo della mappa. All'interno di questo input, si specifica il nodo che contiene l'array di input su cui lo stato demo della mappa deve iterare. Quindi definisci il passaggio che lo stato demo della mappa deve ripetere per generare gli UUID v4. Infine, configuri l'output in modo che ritorni per ogni ripetizione.

1. Scegli il primo stato Pass nel prototipo del tuo flusso di lavoro. Nella scheda Output, inserisci quanto segue in Risultato:

```
{
  "foo": "bar",
  "colors": [
```

```
    "red",
    "green",
    "blue",
    "yellow",
    "white"
  ]
}
```

2. Scegli lo stato dimostrativo della mappa e nella scheda Configurazione, procedi come segue:

   a. Scegli Fornisci un percorso all'array degli elementi.

   b. Specificate il seguente percorso di riferimento per selezionare il nodo che contiene l'array di input:

   ```
   $.colors
   ```

3. Scegliete lo stato Genera UUID e nella scheda Input, effettuate le seguenti operazioni:

   a. Scegliete Trasforma input con parametri.

   b. Inserisci il seguente input JSON per generare gli UUID v4 per ciascuno degli elementi dell'array di input. Si utilizza la funzione `States.UUID` intrinseca per generare gli UUID.

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

4. Per lo stato Genera UUID, scegliete la scheda Output ed effettuate le seguenti operazioni:

   a. Scegliete Filtra output con. OutputPath

   b. Immettete il seguente percorso di riferimento per selezionare il nodo JSON che contiene gli elementi dell'array di output:

   ```
   $.uuid
   ```

Fase 3: rivedi la definizione di Amazon States Language generata automaticamente e salva il flusso di lavoro

Mentre trascini gli stati dal pannello Flow sull'area di lavoro, Workflow Studio compone automaticamente la definizione Amazon States Language (ASL) del tuo flusso di lavoro in tempo reale. Puoi modificare questa definizione in base alle tue esigenze.

1. (Facoltativo) Scegli Definizione nel ✓Inspector pannello per visualizzare la definizione del tuo flusso di lavoro generata automaticamente da Amazon States Language.

   > **Tip**
   >
   > Puoi anche visualizzare la definizione ASL in Workflow Studio. Editor del codice Nell'editor di codice, puoi anche modificare la definizione ASL del tuo flusso di lavoro.

   L'esempio seguente mostra la definizione di Amazon States Language generata automaticamente per il tuo flusso di lavoro.

```
{
    "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. Specificate un nome per la vostra macchina a stati. Per fare ciò, scegli l'icona di modifica accanto al nome della macchina a stati predefinita di MyStateMachine. Quindi, nella configurazione della macchina a stati, specifica un nome nella casella Nome macchina a stati.

   Per questo tutorial, inserisci il nome **`InlineMapDemo`**.

3. (Facoltativo) Nella configurazione della macchina a stati, specificate altre impostazioni del flusso di lavoro, come il tipo di macchina a stati e il relativo ruolo di esecuzione.

   Per questo tutorial, mantieni tutte le selezioni predefinite nella configurazione della macchina a stati.

4. Nella finestra di dialogo Conferma creazione del ruolo, scegli Conferma per continuare.

   Puoi anche scegliere Visualizza le impostazioni del ruolo per tornare alla configurazione della macchina a stati.

> **Note**
>
> Se elimini il ruolo IAM creato da Step Functions, Step Functions non può ricrearlo in un secondo momento. Allo stesso modo, se modifichi il ruolo (ad esempio, rimuovendo

Step Functions dai principi nella policy IAM), Step Functions non può ripristinare le impostazioni originali in un secondo momento.

Fase 4: Esegui la macchina a stati

Le esecuzioni di macchine a stati sono casi in cui si esegue il flusso di lavoro per eseguire attività.

1. Nella InlineMapDemopagina, scegli Avvia esecuzione.
2. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

    1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

        > **Note**
        >
        > Step Functions consente di creare nomi per macchine a stati, esecuzioni, attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

    2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.
    3. Selezionare Start execution (Avvia esecuzione).
    4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

        Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Per visualizzare l'input e l'output dell'esecuzione side-by-side, scegliete Input e output di esecuzione. In Output, visualizza l'array di output restituito dallo `Map` stato. Di seguito è riportato un esempio dell'array di 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"
]
```

Copiare dati CSV su larga scala utilizzando Distributed Map

Questo tutorial ti aiuta a iniziare a usare lo `Map` stato in modalità Distribuita. Uno `Map` stato impostato su Distributed è noto come stato Distributed Map. Utilizzi lo stato Distributed Map nei tuoi flussi di lavoro per iterare su fonti di dati Amazon S3 su larga scala. Lo `Map` stato esegue ogni iterazione come un'esecuzione secondaria del flusso di lavoro, il che consente un'elevata concorrenza. Per ulteriori informazioni sulla modalità distribuita, consulta Map state in Distributed mode.

In questo tutorial, usi lo stato della mappa distribuita per eseguire iterazioni su un file CSV in un bucket Amazon S3. Quindi ne restituisci il contenuto, insieme all'ARN dell'esecuzione di un workflow secondario, in un altro bucket Amazon S3. Inizi creando un prototipo di flusso di lavoro in Workflow Studio. Successivamente, impostate la modalità di elaborazione `Map` dello stato su Distribuito, specificate il file CSV come set di dati e fornite la sua posizione allo stato. `Map` È inoltre necessario specificare il tipo di flusso di lavoro per le esecuzioni secondarie del flusso di lavoro che lo stato Distributed Map avvia come Express.

Oltre a queste impostazioni, si specificano anche altre configurazioni, come il numero massimo di esecuzioni simultanee di flussi di lavoro secondari e la posizione in cui esportare il `Map` risultato, per il flusso di lavoro di esempio utilizzato in questo tutorial.

Indice

- Prerequisiti
- Passaggio 1: crea il prototipo del flusso di lavoro
- Passaggio 2: configura i campi richiesti per lo stato della mappa

- Passaggio 3: Configurare le opzioni aggiuntive
- Fase 4: Configurare la funzione Lambda
- Passaggio 5: aggiorna il prototipo del flusso di lavoro
- Passaggio 6: rivedi la definizione di Amazon States Language generata automaticamente e salva il flusso di lavoro
- Passaggio 7: Esegui la macchina a stati

Prerequisiti

- Carica un file CSV in un bucket Amazon S3. Devi definire una riga di intestazione all'interno del file CSV. Per informazioni sui limiti di dimensione imposti al file CSV e su come specificare la riga di intestazione, consulta. File CSV in un bucket Amazon S3
- Crea un altro bucket Amazon S3 e una cartella all'interno di quel bucket in cui esportare il risultato dello `Map` stato.

> **Important**
>
> Assicurati che i bucket Amazon S3 si trovino nella stessa macchina Account AWS a Regione AWS stati.

Passaggio 1: crea il prototipo del flusso di lavoro

In questo passaggio, crei il prototipo per il tuo flusso di lavoro utilizzando Workflow Studio. Workflow Studio è un designer visivo di flussi di lavoro disponibile nella console Step Functions. È possibile scegliere lo stato e l'azione API richiesti rispettivamente dalle schede Flusso e Azioni. Utilizzerai la funzionalità drag and drop di Workflow Studio per creare il prototipo del flusso di lavoro.

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Nella finestra di dialogo Scegli un modello, seleziona Vuoto.
3. Scegli Seleziona. Questo apre Workflow Studio inModalità di progettazione.
4. Dalla scheda Flusso, trascina uno stato della mappa e rilascialo nello stato vuoto denominato Trascina il primo stato qui.
5. Nella scheda Configurazione, per Nome dello stato, inserisci**`Process data`**.

6. Dalla scheda Azioni, trascina un'azione AWS Lambda Invoke API e rilasciala nello stato dei dati del processo.
7. Rinomina lo stato AWS Lambda Invoke in. **`Process CSV data`**

Passaggio 2: configura i campi richiesti per lo stato della mappa

In questo passaggio, configuri i seguenti campi obbligatori dello stato della mappa distribuita:

- ItemReader— specifica il set di dati e la sua posizione da cui `Map` lo stato può leggere l'input.
- ItemProcessor— specifica i seguenti valori:
  - `ProcessorConfig`— Imposta `EXPRESS` rispettivamente `Mode` e `ExecutionType` su `DISTRIBUTED` e. Questo imposta la modalità di elaborazione `Map` dello stato e il tipo di workflow per le esecuzioni di workflow secondarie avviate dallo stato Distributed Map.
  - `StartAt`— Il primo stato del flusso di lavoro Map.
  - `States`— Definisce il flusso di lavoro Map, che è un insieme di passaggi da ripetere in ogni esecuzione del workflow secondario.
- ResultWriter: specifica la posizione Amazon S3 in cui Step Functions scrive i risultati dello stato della mappa distribuita.

> **Important**
>
> Assicurati che il bucket Amazon S3 che usi per esportare i risultati di un Map Run si trovi nella stessa macchina Account AWS a Regione AWS stati. In caso contrario, l'esecuzione della macchina a stati fallirà con l'`States.ResultWriterFailed`errore.

Per configurare i campi obbligatori:

1. Scegli lo stato dei dati del processo e, nella scheda Configurazione, procedi come segue:
   a. Per la modalità di elaborazione, scegli Distribuito.
   b. Per Origine articolo, scegli Amazon S3, quindi scegli il file CSV in S3 dall'elenco a discesa dei sorgenti degli articoli S3.
   c. Effettua le seguenti operazioni per specificare la posizione Amazon S3 del tuo file CSV:
      i. Per l'oggetto S3, seleziona Inserisci bucket and key dall'elenco a discesa.

ii. Per Bucket, inserisci il nome del bucket Amazon S3, che contiene il file CSV. Ad esempio, **`sourceBucket`**.

iii. Per Key, inserisci il nome dell'oggetto Amazon S3 in cui hai salvato il file CSV. È inoltre necessario specificare il nome del file CSV in questo campo. Ad esempio, **`csvDataset/ratings.csv`**.

d. Per i file CSV, devi anche specificare la posizione dell'intestazione della colonna. Per fare ciò, scegli Configurazione aggiuntiva, quindi per la posizione dell'intestazione CSV mantieni la selezione predefinita di Prima riga se la prima riga del file CSV è l'intestazione. Altrimenti, scegli Dato per specificare l'intestazione all'interno della definizione della macchina a stati. Per ulteriori informazioni, consulta `ReaderConfig`.

e. Per il tipo di esecuzione Child, scegliete Express.

2. In Esporta posizione, per esportare i risultati di Map Run in una posizione Amazon S3 specifica, seleziona Esporta l'output dello stato della mappa su Amazon S3.

3. Esegui questa operazione:

a. Per il bucket S3, scegli Inserisci il nome e il prefisso del bucket dall'elenco a discesa.

b. Per Bucket, inserisci il nome del bucket Amazon S3 in cui desideri esportare i risultati. Ad esempio, **`mapOutputs`**.

c. In Prefix, inserisci il nome della cartella in cui vuoi salvare i risultati. Ad esempio, **`resultData`**.

Passaggio 3: Configurare le opzioni aggiuntive

Oltre alle impostazioni richieste per lo stato di una mappa distribuita, puoi specificare anche altre opzioni. Queste possono includere il numero massimo di esecuzioni simultanee di workflow secondari e la posizione in cui esportare il risultato `Map` dello stato.

1. Scegli lo stato dei dati del processo. Quindi, in Origine dell'articolo, scegli Configurazione aggiuntiva.

2. Esegui questa operazione:

a. Scegli Modifica elementi con ItemSelector per specificare un input JSON personalizzato per ogni esecuzione del flusso di lavoro secondario.

b. Inserisci il seguente input JSON:

```
{
  "index.$": "$$.Map.Item.Index",
  "value.$": "$$.Map.Item.Value"
}
```

Per informazioni su come creare un input personalizzato, consulta`ItemSelector`.

3. Nelle impostazioni di runtime, per Limite di concorrenza, specifica il numero di esecuzioni simultanee di flussi di lavoro secondari che lo stato della mappa distribuita può avviare. Ad esempio, specifica **`100`**.
4. Apri una nuova finestra o scheda nel browser e completa la configurazione della funzione Lambda che utilizzerai in questo flusso di lavoro, come spiegato in. Fase 4: Configurare la funzione Lambda

Fase 4: Configurare la funzione Lambda

> **Important**
>
> Assicurati che la tua funzione Lambda sia utilizzata Regione AWS come la tua macchina a stati.

1. Apri la console Lambda e scegli Crea funzione.
2. Nella pagina Create function (Crea funzione), scegliere Author from scratch (Crea da zero).
3. Nella sezione Informazioni di base, configura la tua funzione Lambda:
   a. Nel campo Function name (Nome funzione), immettere **`distributedMapLambda`**.
   b. Per Runtime scegli Node.js 16.x.
   c. Mantieni tutte le selezioni predefinite e scegli Crea funzione.
   d. Dopo aver creato la funzione Lambda, copia l'Amazon Resource Name (ARN) della funzione visualizzato nell'angolo in alto a destra della pagina. Dovrai inserirlo nel prototipo del tuo flusso di lavoro. Per copiare l'ARN, fate clic su.

   Di seguito è riportato un esempio di ARN:

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

4. Copia il codice seguente per la funzione Lambda e incollalo nella sezione Codice sorgente della distributedMapLambdapagina.

```
exports.handler = async function(event, context) {
    console.log("Received Input:\n", event);

    return {
        'statusCode' : 200,
        'inputReceived' : event //returns the input that it received
    }
};
```

5. Seleziona Deploy (Implementa). Una volta implementata la funzione, scegli Test per vedere l'output della tua funzione Lambda.

Passaggio 5: aggiorna il prototipo del flusso di lavoro

Nella console Step Functions, aggiornerai il tuo flusso di lavoro per aggiungere l'ARN della funzione Lambda.

1. Torna alla scheda o alla finestra in cui hai creato il prototipo del flusso di lavoro.
2. Scegliete la fase Elabora dati CSV e, nella scheda Configurazione, effettuate le seguenti operazioni:

   a. Per Tipo di integrazione, scegli Ottimizzato.

   b. Per Nome funzione, inizia a inserire il nome della tua funzione Lambda. Scegli la funzione dall'elenco a discesa visualizzato oppure scegli Inserisci il nome della funzione e fornisci l'ARN della funzione Lambda.

Passaggio 6: rivedi la definizione di Amazon States Language generata automaticamente e salva il flusso di lavoro

Mentre trascini gli stati dalle schede Action e Flow sull'area di disegno, Workflow Studio compone automaticamente la definizione Amazon States Language del tuo flusso di lavoro in tempo reale. Puoi modificare questa definizione in base alle tue esigenze.

1. (Facoltativo) Scegliete Definizione sul ✓Inspector pannello e visualizzate la definizione della macchina a stati.

> **Tip**
>
> È inoltre possibile visualizzare la definizione ASL in Workflow Studio. Editor del codice Nell'editor di codice, puoi anche modificare la definizione ASL del tuo flusso di lavoro.

Il codice di esempio seguente mostra la definizione di Amazon States Language generata automaticamente per il tuo flusso di lavoro.

```
{
  "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. Specificate un nome per la vostra macchina a stati. Per fare ciò, scegli l'icona di modifica accanto al nome della macchina a stati predefinita di MyStateMachine. Quindi, nella configurazione della macchina a stati, specifica un nome nella casella Nome macchina a stati.

   Per questo tutorial, inserisci il nome **`DistributedMapDemo`**.

3. (Facoltativo) Nella configurazione della macchina a stati, specificate altre impostazioni del flusso di lavoro, come il tipo di macchina a stati e il relativo ruolo di esecuzione.

   Per questo tutorial, mantieni tutte le selezioni predefinite nella configurazione della macchina a stati.

4. Nella finestra di dialogo Conferma creazione del ruolo, scegli Conferma per continuare.

   Puoi anche scegliere Visualizza le impostazioni del ruolo per tornare alla configurazione della macchina a stati.

> **Note**
>
> Se elimini il ruolo IAM creato da Step Functions, Step Functions non può ricrearlo in un secondo momento. Allo stesso modo, se modifichi il ruolo (ad esempio, rimuovendo Step Functions dai principi nella policy IAM), Step Functions non può ripristinare le impostazioni originali in un secondo momento.

Passaggio 7: Esegui la macchina a stati

Un'esecuzione è un'istanza della macchina a stati in cui si esegue il flusso di lavoro per eseguire attività.

1. Nella DistributedMapDemopagina, scegli Avvia esecuzione.
2. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

    1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

        > **Note**
        >
        > Step Functions consente di creare nomi per macchine a stati, esecuzioni, attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

    2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.
    3. Selezionare Start execution (Avvia esecuzione).
    4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

        Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di

ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vedere Pagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Ad esempio, scegli lo `Map` stato, quindi scegli Map Run per aprire la pagina Map Run Details. In questa pagina, è possibile visualizzare tutti i dettagli di esecuzione dello stato della mappa distribuita e delle esecuzioni del workflow secondario avviate. Per informazioni su questa pagina, vedere Esame di Map Run.

Elaborazione di un intero batch di dati con una funzione Lambda

In questo tutorial, usi il ItemBatcher campo dello stato della mappa distribuita per elaborare un intero batch di elementi all'interno di una funzione Lambda. Ogni batch contiene un massimo di tre articoli. Lo stato Distributed Map avvia quattro esecuzioni di workflow secondarie, in cui ogni esecuzione elabora tre elementi, mentre un'esecuzione elabora un singolo elemento. Ogni esecuzione del workflow secondario richiama una funzione Lambda che esegue un'iterazione sui singoli elementi presenti nel batch.

Creerai una macchina a stati che esegue la moltiplicazione su una matrice di numeri interi. Supponiamo che l'array di numeri interi fornito come input sia `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]` e che il fattore di moltiplicazione sia. 7 Quindi, l'array risultante formato dopo aver moltiplicato questi numeri interi per un fattore 7, sarà. `[7, 14, 21, 28, 35, 42, 49, 56, 63, 70]`

Argomenti

- Fase 1: Creare la macchina a stati
- Fase 2: Creare la funzione Lambda
- Passaggio 3: Esegui la macchina a stati

Fase 1: Creare la macchina a stati

In questo passaggio, crei il prototipo di flusso di lavoro della macchina a stati che passa un intero batch di dati alla funzione Lambda che creerai nel passaggio 2.

- Usa la seguente definizione per creare una macchina a stati utilizzando la console Step Functions. Per informazioni sulla creazione di una macchina a stati, Passaggio 1: crea il prototipo del flusso di lavoro consultate il tutorial Guida introduttiva all'uso di Distributed Map state tutorial.

In questa macchina a stati, si definisce uno stato della mappa distribuita che accetta un array di 10 numeri interi come input e lo passa a una funzione Lambda in batch di. 3 La funzione Lambda esegue un'iterazione sui singoli elementi presenti nel batch e restituisce un array di output denominato. `multiplied` L'array di output contiene il risultato della moltiplicazione eseguita sugli elementi passati nell'array di input.

> **Important**
>
> Assicurati di sostituire l'Amazon Resource Name (ARN) della funzione Lambda nel codice seguente con l'ARN della funzione che creerai nella fase 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"
    }
  }
}
```

Fase 2: Creare la funzione Lambda

In questo passaggio, si crea la funzione Lambda che elabora tutti gli elementi passati nel batch.

> **Important**
>
> Assicurati che la tua funzione Lambda sia utilizzata Regione AWS come la tua macchina a stati.

Creazione della funzione Lambda

1. Usa la console Lambda per creare una funzione Lambda di Python 3.9 denominata. **`ProcessEntireBatch`** Per informazioni sulla creazione di una funzione Lambda, consulta Fase 4: Configurazione della funzione Lambda nel tutorial Guida introduttiva all'uso di Distributed Map state.
2. Copia il codice seguente per la funzione Lambda e incollalo nella sezione Codice sorgente della tua funzione Lambda.

```
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. Dopo aver creato la funzione Lambda, copia l'ARN della funzione visualizzato nell'angolo superiore destro della pagina. Per copiare l'ARN, fare clic su.

   Di seguito è riportato un esempio di ARN, dove *`function-name`*è il nome della funzione Lambda (in questo caso,): `ProcessEntireBatch`

```
arn:aws:lambda:us-east-1:123456789012:function:function-name
```

   Dovrai fornire la funzione ARN nella macchina a stati creata nel passaggio 1.
4. Scegli Deploy per distribuire le modifiche.

Passaggio 3: Esegui la macchina a stati

Quando si esegue la macchina a stati, lo stato Distributed Map avvia quattro esecuzioni di workflow secondarie, in cui ogni esecuzione elabora tre elementi, mentre un'esecuzione elabora un singolo elemento.

L'esempio seguente mostra i dati passati alla `ProcessEntireBatch`funzione da una delle esecuzioni secondarie del flusso di lavoro.

```
{
  "BatchInput": {
    "MyMultiplicationFactor": 7
  },
  "Items": [1, 2, 3]
}
```

Dato questo input, l'esempio seguente mostra l'array di output denominato `multiplied` restituito dalla funzione Lambda.

```
{
  "statusCode": 200,
  "multiplied": [7, 14, 21]
}
```

La macchina a stati restituisce il seguente output che contiene quattro array denominati `multiplied` per le quattro esecuzioni dei flussi di lavoro secondari. Questi array contengono i risultati della moltiplicazione dei singoli elementi di input.

```
[
  {
    "statusCode": 200,
    "multiplied": [7, 14, 21]
  },
  {
    "statusCode": 200,
    "multiplied": [28, 35, 42]
  },
  {
    "statusCode": 200,
    "multiplied": [49, 56, 63]
  },
  {
    "statusCode": 200,
    "multiplied": [70]
  }
]
```

Per combinare tutti gli elementi dell'array restituiti in un unico array di output, puoi usare il campo. ResultSelector Definisci questo campo all'interno dello stato della mappa distribuita per trovare tutti gli `multiplied` array, estrarre tutti gli elementi all'interno di questi array e quindi combinarli in un unico array di output.

Per utilizzare il `ResultSelector` campo, aggiorna la definizione della macchina a stati come mostrato nell'esempio seguente.

```
{
  "StartAt": "Pass",
  "States": {
    ...
    ...
    "Map": {
      "Type": "Map",
      ...
      ...
      "ItemsPath": "$.MyItems",
      "ResultSelector": {
        "multiplied.$": "$..multiplied[*]"
      }
    }
  }
}
```

La macchina a stati aggiornata restituisce un array di output consolidato come mostrato nell'esempio seguente.

```
{
  "multiplied": [7, 14, 21, 28, 35, 42, 49, 56, 63, 70]
}
```

Elaborazione di singoli elementi di dati con una funzione Lambda

In questo tutorial, si utilizza il ItemBatcher campo dello stato della mappa distribuita per iterare sui singoli elementi presenti in un batch utilizzando una funzione Lambda. Lo stato Distributed Map avvia quattro esecuzioni di workflow secondarie. Ciascuno di questi flussi di lavoro secondari esegue uno stato della mappa in linea. Per ogni iterazione, lo stato Inline Map richiama una funzione Lambda e passa un singolo elemento dal batch alla funzione. La funzione Lambda elabora quindi l'elemento e restituisce il risultato.

Creerai una macchina a stati che esegue la moltiplicazione su una matrice di numeri interi. Supponiamo che l'array di numeri interi fornito come input sia `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]` e che il fattore di moltiplicazione sia. 7 Quindi, l'array risultante formato dopo aver moltiplicato questi numeri interi per un fattore 7, sarà. `[7, 14, 21, 28, 35, 42, 49, 56, 63, 70]`

Argomenti

- Fase 1: Creare la macchina a stati
- Fase 2: Creare la funzione Lambda
- Passaggio 3: Esegui la macchina a stati

Fase 1: Creare la macchina a stati

In questo passaggio, crei il prototipo di flusso di lavoro della macchina a stati che passa un singolo elemento da un batch di elementi a ogni invocazione della funzione Lambda che creerai nel Passaggio 2.

- Usa la seguente definizione per creare una macchina a stati utilizzando la console Step Functions. Per informazioni sulla creazione di una macchina a stati, Passaggio 1: crea il prototipo del flusso di lavoro consultate il tutorial Guida introduttiva all'uso di Distributed Map state tutorial.

  In questa macchina a stati, si definisce uno stato della mappa distribuita che accetta un array di 10 numeri interi come input e passa questi elementi dell'array alle esecuzioni secondarie del flusso di lavoro in batch. Ogni esecuzione del workflow secondario riceve un batch di tre elementi come input ed esegue uno stato Inline Map. Ogni iterazione dello stato Inline Map richiama una funzione Lambda e passa un elemento dal batch alla funzione. Questa funzione quindi moltiplica l'elemento per un fattore di e restituisce il risultato. 7

  L'output di ogni esecuzione del workflow secondario è un array JSON che contiene il risultato della moltiplicazione per ciascuno degli elementi passati.

  > **Important**
  >
  > Assicurati di sostituire l'Amazon Resource Name (ARN) della funzione Lambda nel codice seguente con l'ARN della funzione che creerai nella fase 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"
      }
    }
  }
}
```

Fase 2: Creare la funzione Lambda

In questo passaggio, si crea la funzione Lambda che elabora ogni elemento passato dal batch.

> **Important**
>
> Assicurati che la tua funzione Lambda sia utilizzata Regione AWS come la tua macchina a stati.

Creazione della funzione Lambda

1. Usa la console Lambda per creare una funzione Lambda di Python 3.9 denominata. **`ProcessSingleItem`** Per informazioni sulla creazione di una funzione Lambda, consulta Fase 4: Configurazione della funzione Lambda nel tutorial Guida introduttiva all'uso di Distributed Map state.
2. Copia il codice seguente per la funzione Lambda e incollalo nella sezione Codice sorgente della tua funzione Lambda.

```
import json

def lambda_handler(event, context):

    multiplication_factor = event['MyMultiplicationFactor']
    item = event['MyItem']

    result = multiplication_factor * item

    return {
        'statusCode': 200,
        'multiplied': result
    }
```

3. Dopo aver creato la funzione Lambda, copia l'ARN della funzione visualizzato nell'angolo superiore destro della pagina. Per copiare l'ARN, fare clic su.

   Di seguito è riportato un esempio di ARN, dove *`function-name`*è il nome della funzione Lambda (in questo caso,): `ProcessSingleItem`

   ```
   arn:aws:lambda:us-east-1:123456789012:function:function-name
   ```

   Dovrai fornire la funzione ARN nella macchina a stati creata nel passaggio 1.
4. Scegli Deploy per distribuire le modifiche.

Passaggio 3: Esegui la macchina a stati

Quando si esegue la macchina a stati, lo stato Distributed Map avvia quattro esecuzioni di workflow secondarie, in cui ogni esecuzione elabora tre elementi, mentre un'esecuzione elabora un singolo elemento.

L'esempio seguente mostra i dati passati a una delle chiamate di `ProcessSingleItem`funzione all'interno di un'esecuzione di workflow secondario.

```
{
  "MyMultiplicationFactor": 7,
  "MyItem": 1
}
```

Dato questo input, l'esempio seguente mostra l'output restituito dalla funzione Lambda.

```
{
  "statusCode": 200,
  "multiplied": 7
}
```

L'esempio seguente mostra l'array JSON di output per una delle esecuzioni secondarie del flusso di lavoro.

```
[
  {
    "statusCode": 200,
    "multiplied": 7
  },
  {
    "statusCode": 200,
    "multiplied": 14
  },
  {
    "statusCode": 200,
    "multiplied": 21
  }
]
```

La macchina a stati restituisce il seguente output che contiene quattro array per le quattro esecuzioni dei flussi di lavoro secondari. Questi array contengono i risultati della moltiplicazione dei singoli elementi di input.

Infine, l'output della macchina a stati è un array denominato `multiplied` che combina tutti i risultati di moltiplicazione restituiti per le quattro esecuzioni dei flussi di lavoro secondari.


```
[
  [
    {
      "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
    }
  ]
]
```

Per combinare tutti i risultati di moltiplicazione restituiti dalle esecuzioni dei flussi di lavoro secondari in un unico array di output, puoi utilizzare il campo. ResultSelector Definite questo campo all'interno dello stato della Mappa distribuita per trovare tutti i risultati, estrarre i singoli risultati e quindi combinarli in un unico array di output denominato. `multiplied`

Per utilizzare il `ResultSelector` campo, aggiorna la definizione della macchina a stati come mostrato nell'esempio seguente.

```
{
  "StartAt": "Pass",
  "States": {
    ...
    ...
    "Map": {
      "Type": "Map",
      ...
      ...
      "ItemBatcher": {
        "MaxItemsPerBatch": 3,
        "BatchInput": {
          "MyMultiplicationFactor.$": "$.MyMultiplicationFactor"
        }
      },
      "ItemsPath": "$.MyItems",
      "ResultSelector": {
        "multiplied.$": "$..multiplied"
      }
    }
```

```
  }
}
```

La macchina a stati aggiornata restituisce un array di output consolidato come mostrato nell'esempio seguente.

```
{
  "multiplied": [7, 14, 21, 28, 35, 42, 49, 56, 63, 70]
}
```

Avvio dell'esecuzione di una macchina a stati in risposta agli eventi di Amazon S3

Puoi eseguire una macchina a AWS Step Functions stati in risposta a una EventBridge regola Amazon.

Questo tutorial mostra come configurare una macchina a stati come destinazione per una EventBridge regola Amazon. Questa regola avvierà l'esecuzione di una macchina a stati quando i file vengono aggiunti a un bucket Amazon Simple Storage Service (Amazon S3).

Per un'applicazione pratica, puoi avviare una macchina a stati che esegue operazioni sui file che aggiungi al bucket, come la creazione di miniature o l'esecuzione dell'analisi di Amazon Rekognition su file di immagini e video.

In questo tutorial, inizierai l'esecuzione di una macchina a `Helloworld` stati caricando un file in un bucket Amazon S3. Quindi esamini l'input di esempio di tale esecuzione per identificare le informazioni incluse nell'input della notifica di evento di Amazon S3 inviata a. EventBridge

Argomenti

- Prerequisito: creazione di una macchina a stati
- Fase 1: creare un bucket in Amazon S3
- Fase 2: abilitare la notifica degli eventi di Amazon S3 con EventBridge
- Passaggio 3: crea una EventBridge regola Amazon
- Fase 4: test della regola
- Esempio di input di esecuzione

Prerequisito: creazione di una macchina a stati

Prima di poter configurare una macchina a stati come EventBridge destinazione Amazon, devi creare la macchina a stati.

- Per creare una macchina a stati di base, usa il tutorial Creazione di una macchina a stati che utilizza una funzione Lambda.
- Se disponi già di una macchina a stati `Helloworld`, vai alla fase successiva.

Fase 1: creare un bucket in Amazon S3

Ora che hai una macchina a `Helloworld` stati, devi creare un bucket Amazon S3 per archiviare i tuoi file. Nella fase 3 di questo tutorial, imposti una regola in modo che quando un file viene caricato in questo bucket, attivi l'esecuzione della tua EventBridge macchina a stati.

1. Accedi alla console Amazon S3, quindi scegli Crea bucket per creare il bucket in cui desideri archiviare i file e attivare una regola di evento Amazon S3.
2. Immettere Bucket name (Nome bucket), come *`username`*`-sfn-tutorial`.

   > **Note**
   >
   > I nomi dei bucket devono essere univoci per tutti i nomi di bucket esistenti in tutte le AWS regioni di Amazon S3. Utilizzare il proprio *`nome utente`* per rendere questo nome univoco. È necessario creare tutte le risorse nella stessa regione AWS.

3. Mantieni tutte le selezioni predefinite sulla pagina e scegli Crea bucket.

Fase 2: abilitare la notifica degli eventi di Amazon S3 con EventBridge

Dopo aver creato il bucket Amazon S3, configuralo per inviare eventi EventBridge ogni volta che si verificano determinati eventi nel bucket S3, come i caricamenti di file.

1. Accedi alla console Amazon S3.
2. Nell'elenco Bucket (Bucket) scegliere il nome del bucket per il quale abilitare gli eventi.
3. Scegliere Properties (Proprietà).
4. Scorri la pagina verso il basso per visualizzare la sezione Notifiche sugli eventi, quindi scegli Modifica nella EventBridge sottosezione Amazon.

5. In Invia notifiche ad Amazon EventBridge per tutti gli eventi in questo bucket, scegli Attivato.
6. Sceglie Save changes (Salva modifiche).

> **Note**
>
> Dopo l'attivazione EventBridge, occorrono circa cinque minuti prima che le modifiche abbiano effetto.

Passaggio 3: crea una EventBridge regola Amazon

Dopo aver creato una macchina a stati e aver creato il bucket Amazon S3 e averlo configurato per inviare notifiche di eventi a EventBridge, crea una regola. EventBridge

> **Note**
>
> È necessario configurare la EventBridge regola nella stessa AWS regione del bucket Amazon S3.

Per creare la regola

1. Vai alla EventBridge console Amazon, scegli Crea regola.

> **Tip**
>
> In alternativa, nel riquadro di navigazione della EventBridge console, scegli Regole in Autobus, quindi scegli Crea regola.

2. Inserisci un nome per la regola (ad esempio`S3Step Functions`) e, facoltativamente, inserisci una descrizione per la regola.
3. Per Event bus e Rule type, mantieni le selezioni predefinite.
4. Seleziona Successivo. Si apre la pagina Build event pattern.
5. Scorri verso il basso fino alla sezione Event pattern ed esegui le seguenti operazioni:

   a. Per Event source, mantieni la selezione predefinita di AWSeventi o eventi EventBridge partner.

b. Per l'AWSassistenza, scegli Simple Storage Service (S3).

c. Per il tipo di evento, scegli Amazon S3 Event Notification.

d. Scegli Evento/i specifico/i, quindi scegli Oggetto creato.

e. Scegliete uno o più bucket specifici per nome e inserite il nome del bucket creato nel passaggio 1 (*username*`-sfn-tutorial`) per archiviare i file.

f. Seleziona Successivo. Si apre la pagina Seleziona obiettivi.

Per creare il target

1. In Target 1, mantieni la selezione predefinita del AWSservizio.
2. Nell'elenco a discesa Seleziona una destinazione, seleziona Step Functions state machine.
3. Nell'elenco Macchina a stati, seleziona la macchina a stati creata in precedenza (ad esempio,`Helloworld`).
4. Mantieni tutte le selezioni predefinite nella pagina e scegli Avanti. Si apre la pagina Configura i tag.
5. Scegliere Next (Successivo) di nuovo. Si apre la pagina Rivedi e crea.
6. Rivedi i dettagli della regola e scegli Create rule (Crea regola).

   La regola viene creata e viene visualizzata la pagina Regole, che elenca tutte le EventBridge regole Amazon.

Fase 4: test della regola

Ora che tutto è a posto, prova ad aggiungere un file al bucket Amazon S3, quindi esamina l'input dell'esecuzione della macchina a stati risultante.

1. Aggiungi un file al tuo bucket Amazon S3.

   Accedi alla console Amazon S3, scegli il bucket che hai creato per archiviare i file (***username*`-sfn-tutorial`**), quindi scegli Carica.
2. Aggiungi un file, ad esempio***test.png***, quindi scegli Carica.

   In questo modo, viene avviata un'esecuzione della macchina a stati, trasmettendo informazioni da AWS CloudTrail come input.
3. Controllare l'esecuzione della macchina a stati.

Vai alla console Step Functions e seleziona la macchina a stati utilizzata nella tua EventBridge regola Amazon (`Helloworld`).

4. Seleziona l'esecuzione più recente di quella macchina a stati ed espandi la sezione Execution Input.

   Questo input include informazioni come il nome del bucket e il nome dell'oggetto. In un caso d'uso reale, una macchina a stati può utilizzare questo input per eseguire azioni su quell'oggetto.

Esempio di input di esecuzione

L'esempio seguente mostra un input tipico per l'esecuzione della macchina a stati.

```
{
    "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"
    }
  }
```

Creazione di un'API Step Functions utilizzando API Gateway

Puoi utilizzare Amazon API Gateway per associare le tue AWS Step Functions API ai metodi in un'API API Gateway. Quando una richiesta HTTPS viene inviata a un metodo API, API Gateway richiama le tue azioni API Step Functions.

Questo tutorial descrive come creare un'API che utilizza una risorsa e il metodo `POST` per comunicare con l'operazione API `StartExecution`. Utilizzerai la console AWS Identity and Access Management (IAM) per creare un ruolo per API Gateway. Quindi, utilizzerai la console API Gateway per creare un'API API Gateway, creare una risorsa e un metodo e mappare il metodo all'azione `StartExecution` API. Infine, distribuirai e testerai le tue API.

> **Note**
>
> Sebbene Amazon API Gateway possa avviare un'esecuzione di Step Functions chiamando`StartExecution`, è necessario `DescribeExecution` chiamare per ottenere il risultato.

Argomenti

- Fase 1: creazione di un ruolo IAM per API Gateway
- Fase 2: Crea la tua API API Gateway
- Fase 3: Test e implementazione dell'API API Gateway

Fase 1: creazione di un ruolo IAM per API Gateway

Prima di creare l'API API Gateway, devi autorizzare API Gateway a richiamare le azioni API di Step Functions.

Per configurare le autorizzazioni per API Gateway

1. Accedi alla console IAM e scegli Ruoli, Crea ruolo.
2. Nella pagina Seleziona un'entità attendibile, esegui le operazioni seguenti:
   a. Per il tipo di entità affidabile, mantieni la selezione predefinita di Servizio AWS.
   b. Per Use case, scegli API Gateway dall'elenco a discesa.
3. Seleziona API Gateway, quindi scegli Avanti.

4. Nella pagina Add permissions (Aggiungi autorizzazioni), scegli Next (Successivo).
5. (Facoltativo) Nella pagina Nome, revisione e creazione, inserisci i dettagli, ad esempio il nome del ruolo. Ad esempio, specifica **`APIGatewayToStepFunctions`**.
6. Scegli Crea ruolo.

   Il ruolo IAM viene visualizzato nell'elenco dei ruoli.
7. Scegliere il nome del ruolo e prendere nota dell'ARN del ruolo, come illustrato nell'esempio seguente.

   ```
   arn:aws:iam::123456789012:role/APIGatewayToStepFunctions
   ```

Per allegare una policy al ruolo IAM

1. Nella pagina Roles (Ruoli) cerca il ruolo (`APIGatewayToStepFunctions`) e quindi selezionalo.
2. Nella scheda Autorizzazioni, scegli Aggiungi autorizzazioni, quindi scegli Allega politiche.
3. Nella pagina Allega policy, cerca`AWSStepFunctionsFullAccess`, scegli la policy, quindi scegli Aggiungi autorizzazioni.

Fase 2: Crea la tua API API Gateway

Dopo aver creato il tuo ruolo IAM, puoi creare la tua API API Gateway personalizzata.

Per creare l'API

1. Apri la console Amazon API Gateway, quindi scegli Crea API.
2. Nella pagina Scegli un tipo di API, nel riquadro API REST, scegli Build.
3. Nella pagina Crea API REST, seleziona Nuova API, quindi inserisci ***`StartExecutionAPI`*** per il nome dell'API.
4. Mantieni il tipo di endpoint dell'API su Regionale, quindi scegli Crea API.

Per creare una risorsa

1. Nella pagina Risorse dell'***`StartExecutionAPI`***, scegli Crea risorsa.
2. Nella pagina Crea risorsa, inserisci **`execution`** il nome della risorsa, quindi scegli Crea risorsa.

Per creare un metodo POST

1. Scegli la risorsa /execution, quindi scegli il metodo Create.
2. Per Tipo di metodo, sceglietePOST.
3. Per Tipo di integrazione, scegli AWS servizio.
4. Per Regione AWS, scegli una regione dall'elenco.

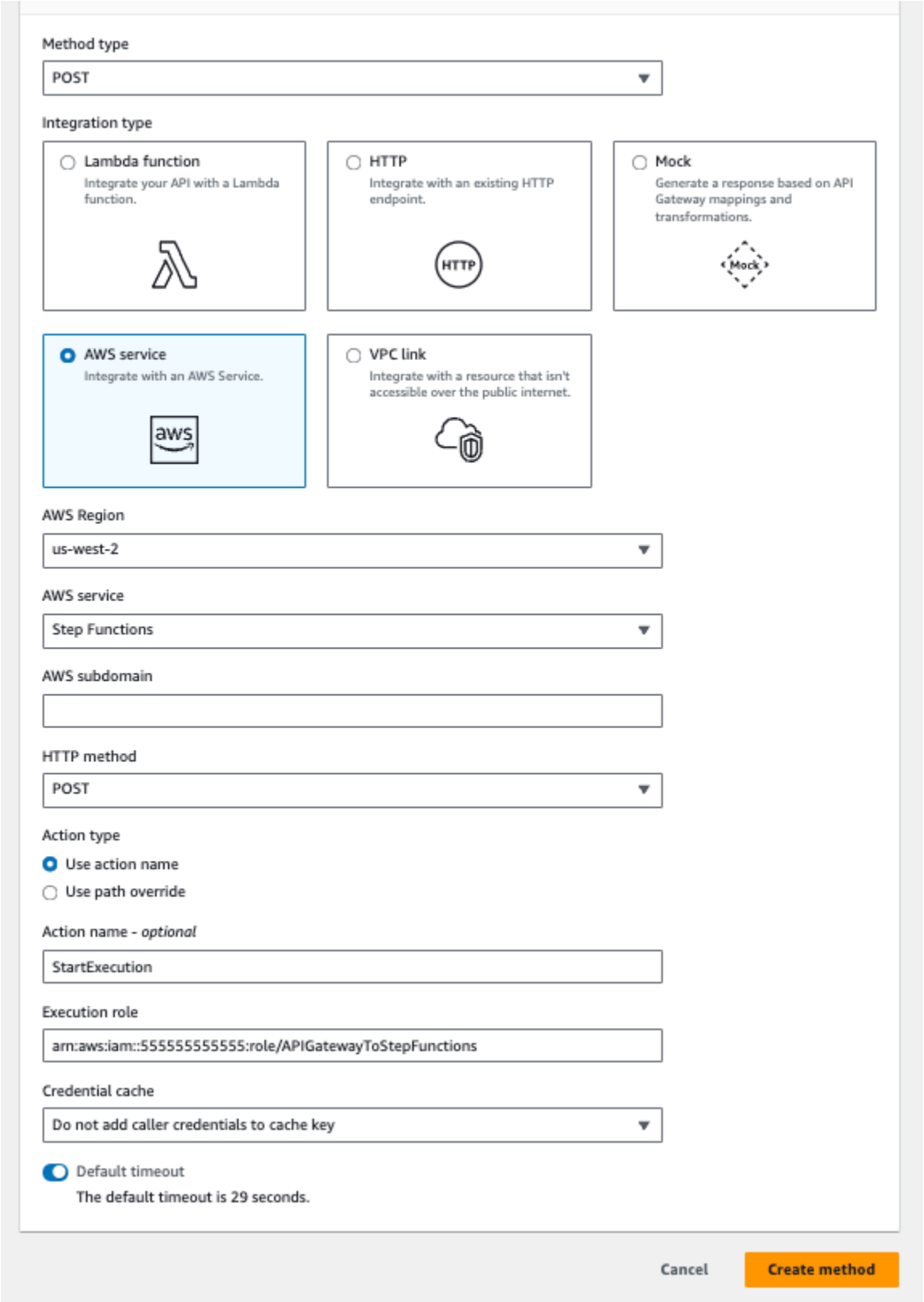
> **Note**
>
> Per le regioni che attualmente supportano Step Functions, vedi Regioni supportate.

5. Per Servizio AWS, scegli Step Functions dall'elenco.
6. Lascia vuoto Sottodominio AWS .
7. Per il metodo HTTP, scegli POST dall'elenco.

> **Note**
>
> Tutte le azioni dell'API Step Functions utilizzano il POST metodo HTTP.

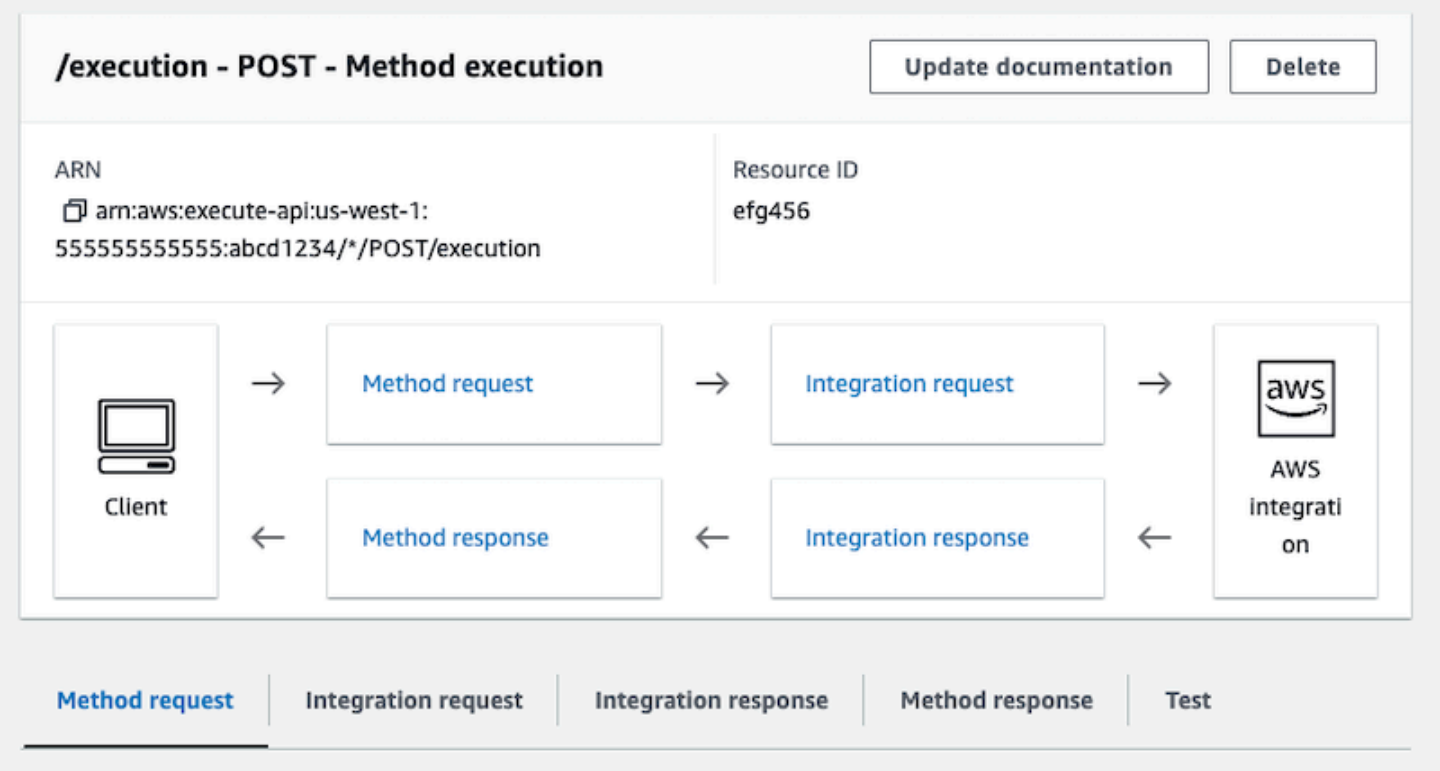
8. Per Tipo di operazione scegli Usa nome operazione.
9. Per Nome azione immetti **`StartExecution`**.
10. Per il ruolo Execution, inserisci il ruolo ARN del ruolo IAM che hai creato in precedenza, come mostrato nell'esempio seguente.

```
arn:aws:iam::123456789012:role/APIGatewayToStepFunctions
```



11. Mantieni le opzioni predefinite per Credential cache e Default timeout, quindi scegli Salva.

La mappatura visiva tra API Gateway e Step Functions viene visualizzata nella pagina /execution - POST - Method execution.



Fase 3: Test e implementazione dell'API API Gateway

Una volta creata l'API, eseguirne i test e la distribuzione.

Per testare la comunicazione tra API Gateway e Step Functions

1. Nella pagina /execution - POST - Method Execution, scegli la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. Nella scheda /execution - POST - Method Test, copia i seguenti parametri di richiesta nella sezione Request body utilizzando l'ARN di una macchina a stati esistente (o crea una nuova macchina a stati che utilizza una funzione Lambda), quindi scegli Test.

```
{
   "input": "{}",
   "name": "MyExecution",
   "stateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:HelloWorld"
}
```

Per ulteriori informazioni, consulta la sintassi della *StartExecution* richiesta nell'API Reference. AWS Step Functions

> **Note**
>
> Se non desideri includere l'ARN della tua macchina a stati nel corpo della chiamata API Gateway, puoi configurare un modello di mappatura nella scheda Richiesta di integrazione, come mostrato nell'esempio seguente.
>
> ```
> {
>     "input": "$util.escapeJavaScript($input.json('$'))",
>     "stateMachineArn": "$util.escapeJavaScript($stageVariables.arn)"
> }
> ```
>
> Con questo approccio, è possibile specificare gli ARN di diverse macchine a stati in base alla fase di sviluppo (ad esempio `devtest`, e). `prod` Per ulteriori informazioni sulla specificazione delle variabili di fase in un modello di mappatura, consulta `$stageVariables`la Guida per sviluppatori di API Gateway.

3. L'esecuzione viene avviata e l'ARN di esecuzione e la relativa data di epoca vengono visualizzati nel corpo della risposta.

```
{
   "executionArn": "arn:aws:states:us-
east-1:123456789012:execution:HelloWorld:MyExecution",
   "startDate": 1486768956.878
}
```

> **Note**
>
> È possibile visualizzare l'esecuzione scegliendo la macchina a stati nella console AWS Step Functions.

Per distribuire l'API

1. Nella pagina Risorse dell'***StartExecutionAPI***, scegli Deploy API.

2. In Fase, seleziona Nuova fase.
3. In Stage name (Nome fase) immettere **`alpha`**.
4. (Facoltativo) In Description (Descrizione), immettere una descrizione.
5. Seleziona Deploy (Implementa).

Per testare la distribuzione

1. Nella pagina Stages dell'***`StartExecutionAPI`***, espandi alpha,/, /execution, POST, quindi scegli il metodo POST.
2. In Method overrides, scegli l'icona di copia per copiare l'URL di richiamo dell'API. L'URL completo dovrebbe essere simile all'esempio seguente.

   ```
   https://a1b2c3d4e5.execute-api.us-east-1.amazonaws.com/alpha/execution
   ```

3. Dalla riga di comando, eseguire il comando `curl` utilizzando l'ARN della macchina a stati, quindi richiamare l'URL della distribuzione, come indicato nell'esempio seguente.

   ```
   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
   ```

   Vengono restituiti l'ARN di esecuzione e la relativa data epoca (Unix epoch), come indicato nell'esempio seguente.

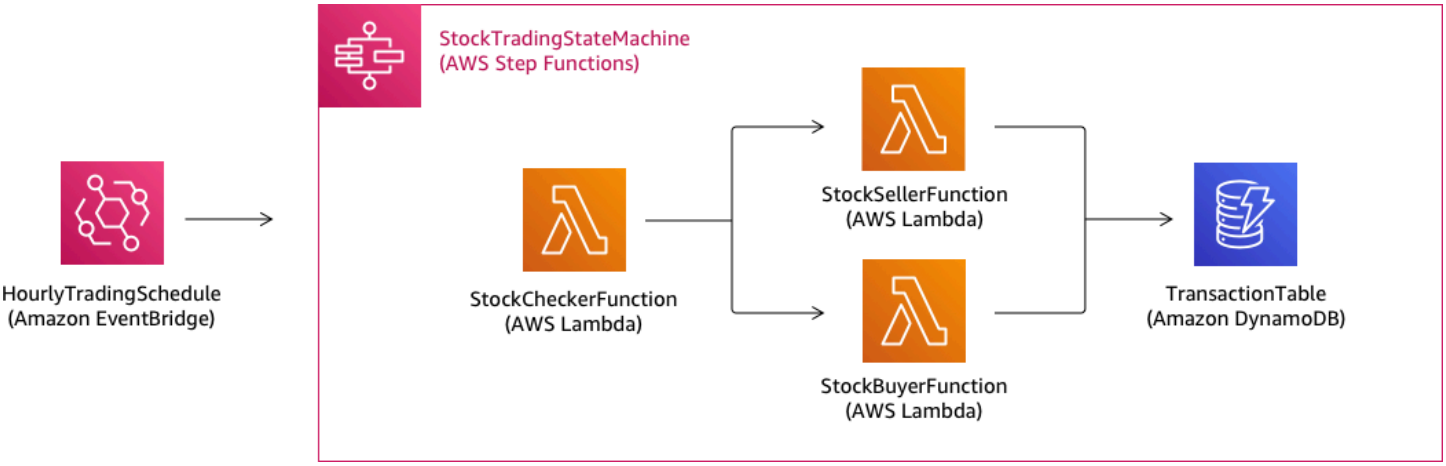
   ```
   {"executionArn":"arn:aws:states:us-
   east-1:123456789012:execution:HelloWorld:MyExecution","startDate":1.486772644911E9}
   ```

   > **Note**
   >
   > Se ricevi l'errore «Token di autenticazione mancante», assicurati che l'URL di invoke termini con /esecuzione.

Creazione di una macchina a stati Step FunctionsAWS SAM

In questa guida è possibile scaricare, compilare e distribuire un'applicazione AWS SAM di esempio contenente una macchina a stati AWS Step Functions. Questa applicazione crea un flusso di lavoro di trading di azioni finte che viene eseguito in base a una pianificazione predefinita (si noti che il programma è disabilitato per impostazione predefinita per evitare addebiti).

Il diagramma seguente mostra i componenti di questa applicazione:



Di seguito è riportata un'anteprima dei comandi eseguiti per creare l'applicazione di esempio. Per ulteriori dettagli su ciascuno di questi comandi, vedere le sezioni più avanti in questa pagina

```
# 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
```

Prerequisiti

Questa guida presuppone che siano stati completati i passaggi descritti nell' Installazione dell'interfaccia a riga di comando AWS SAM per il sistema operativo. Si presuppone che tu abbia fatto quanto segue:

1. Creato unAWSconto.
2. Autorizzazioni IAM configurate.
3. Installato Homebrew. Nota: Homebrew è solo un prerequisito per Linux e macOS.
4. Installato l'interfaccia a riga di comando AWS SAM. Nota: assicurarsi di avere la versione 0.52.0 o successiva. È possibile verificare quale versione si dispone eseguendo il comando `sam --version`.

Passaggio 1: scaricare un'applicazione AWS SAM di esempio

Comando da eseguire:

```
sam init
```

Seguire le istruzioni visualizzate sullo schermo per selezionare quanto segue:

1. Modello: AWSModelli di avvio rapido
2. Lingua: Python, Ruby, NodeJS, Go, Java o .NET
3. Nome progetto: (nome a tua scelta - predefinito è `sam-app`)
4. Fase Quick StartFlusso di lavoro in più fasi

CosaAWS SAMsta facendo:

Questo comando crea una directory con il nome fornito per il prompt 'Nome progetto' (il valore predefinito è `sam-app`). Il contenuto specifico della directory dipenderà dalla lingua scelta.

Di seguito sono riportati i contenuti della directory quando si sceglie uno dei runtime 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
```

Ci sono due file particolarmente interessanti a cui puoi dare un'occhiata:

- `template.yaml`: ContieneAWS SAMmodello che definisce la tua applicazioneAWSrisorse.
- `statemachine/stockTrader.asl.json`: contiene la definizione della macchina a stati dell'applicazione, scritta in Amazon States Language.

È possibile visualizzare la seguente voce nel file `template.yaml`, che punta al file di definizione della macchina a stati:

```
    Properties:
      DefinitionUri: statemachine/stock_trader.asl.json
```

Può essere utile mantenere la definizione della macchina a stati come file separato anziché incorporarla nelAWS SAMmodello. Ad esempio, tenere traccia delle modifiche alla definizione della macchina a stati è più semplice se non si include la definizione nel modello. Puoi utilizzare Workflow Studio per creare e gestire la definizione della macchina a stati ed esportare la definizione dalla console direttamente nel file delle specifiche di Amazon States Language senza unirla al modello.

Per ulteriori informazioni sull'applicazione di esempio, vedere il file `README.md` nella directory del progetto.

Passaggio 2: crea la tua applicazione

Comando da eseguire:

Per prima cosa passare alla directory del progetto (ovvero alla directory in cui si trova il file `template.yaml` per l'applicazione di esempio; per impostazione predefinita è `sam-app`), quindi eseguire questo comando:

```
sam build
```

Esempio di output:

```
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
```

CosaAWS SAMsta facendo:

IlAWS SAMLa CLI include astrazioni per una serie di runtime Lambda per creare le dipendenze e copia tutti gli elementi di compilazione in cartelle di staging in modo che tutto sia pronto per essere impacchettato e distribuito. Il comando `sam build` crea tutte le dipendenze dell'applicazione e copia gli artifact di compilazione nelle cartelle in `.aws-sam/build`.

Fase 3: Caricamento dell'applicazione inAWSCloud

Comando da eseguire:

```
sam deploy --guided
```

Seguire le istruzioni visualizzate sullo schermo. È sufficiente rispondere con `Enter` per accettare le opzioni predefinite fornite nell'esperienza interattiva.

CosaAWS SAMsta facendo:

Questo comando distribuisce l'applicazione inAWSnuvola. Sono necessari gli artefatti di implementazione che crei con`sam build`comando, caricamento dei pacchetti e caricamento in

un bucket Amazon S3AWS SAMCLI e distribuisce l'applicazione utilizzandoAWS CloudFormation. Nell'output del comando di distribuzione puoi vedere le modifiche apportate allo stack AWS CloudFormation.

È possibile verificare che la macchina a stati Step Functions di esempio sia stata distribuita correttamente seguendo questi passaggi:

1. Acceda alAWS Management Consolee apri la console Step Functions all'indirizzohttps://console.aws.amazon.com/states/.
2. Nella navigazione a sinistra, scegliere Macchine a stati.
3. Individuare e scegliere la nuova macchina a stati nell'elenco. Verrà chiamato StockTradingStateMachine-*<unique-hash>*.
4. Scegliere la scheda Definizione .

A questo punto dovrebbe essere visualizzata una rappresentazione visiva della macchina a stati. È possibile verificare che la rappresentazione visiva corrisponda alla definizione della macchina a stati trovata nel file `statemachine/stockTrader.asl.json` della directory del progetto.

Risoluzione dei problemi

Errore CLI SAM: “no such option: --guided"

Durante l'esecuzione di `sam deploy`, viene visualizzato il seguente errore:

```
Error: no such option: --guided
```

Ciò significa che si sta utilizzando una versione precedente dell’interfaccia a riga di comando AWS SAM che non supporta il parametro `--guided`. Per risolvere questo problema, è possibile aggiornare la versione dell’interfaccia a riga di comando AWS SAM alla 0.33.0 o successiva oppure omettere il parametro `--guided` dal comando `sam deploy`.

Errore CLI SAM: “Failed to create managed resources: Unable to locate credentials”

Durante l'esecuzione di `sam deploy`, viene visualizzato il seguente errore:

```
Error: Failed to create managed resources: Unable to locate credentials
```

Ciò significa che non l'hai configuratoAWScredenziali per abilitare ilAWS SAMCLI da creareAWSchiamate di servizio. Per risolvere questo problema, è necessarioAWScredenziali. Per ulteriori informazioni, consulta la paginaConfigurazioneAWSCredenzialinelAWS Serverless Application ModelGuida per gli sviluppatori.

Eliminare

Se non hai più bisogno delAWSle risorse che hai creato eseguendo questo tutorial, puoi rimuoverle eliminando ilAWS CloudFormationpila che hai distribuito.

Per eliminare lo stack AWS CloudFormation creato con questa esercitazione utilizzando AWS Management Console, attenersi alla seguente procedura:

1. Accedere alla AWS Management Console e aprire la console di AWS CloudFormation all'indirizzo https://console.aws.amazon.com/cloudformation.
2. Nel riquadro di navigazione a sinistra, selezionare Stacks (Stack).
3. Nell'elenco degli stack, scegliere sam-app (o il nome dello stack creato).
4. Scegliere Elimina.

Al termine, lo stato dello stack cambierà in DELETE_COMPLETE.

In alternativa, è possibile eliminare lo stack AWS CloudFormation eseguendo il seguente comando AWS CLI:

```
aws cloudformation delete-stack --stack-name sam-app --region region
```

Verifica dello stack eliminato

Per entrambi i metodi di eliminazione diAWS CloudFormationstack, puoi verificare che sia stato eliminato andando suhttps://console.aws.amazon.com/cloudformation, scegliendoPilenel riquadro di navigazione a sinistra e scegliendoEliminatonel menu a discesa a destra della casella di testo per la ricerca. Dovresti vedere il nome dello stack sam-app (o il nome dello stack creato) nell'elenco degli stack eliminati.

Creazione di una macchina a stati Activity utilizzando Step Functions

Questo tutorial descrive come creare una macchina a stati basata su attività mediante Java e AWS Step Functions. Le attività consentono di controllare il codice di lavoro che viene eseguito in un'altra posizione rispetto alla macchina a stati. Per una panoramica, consultare Attività in Come funziona Step Functions.

Per completare questo tutorial, è necessario quanto segue:

- SDK per Java. L'attività di esempio in questo tutorial è un'applicazione Java che utilizza il AWS SDK for Java per comunicare AWS.
- AWS credenziali nell'ambiente o nel file di AWS configurazione standard. Per ulteriori informazioni, consulta Configurare le AWS credenziali nella Guida per gli AWS SDK for Java sviluppatori.

Argomenti

- Fase 1: creazione di un'attività
- Passaggio 2: crea una macchina a stati
- Fase 3: implementazione di un lavoratore
- Fase 4: Eseguire la macchina a stati
- Fase 5: esecuzione e arresto del lavoratore

Fase 1: creazione di un'attività

È necessario rendere Step Functions consapevole dell'attività di cui si desidera creare il lavoratore (un programma). Step Functions risponde con un Amazon Resource Name (ARN) che stabilisce un'identità per l'attività. Utilizza questa identità per coordinare le informazioni trasmesse tra la macchina a stati e il lavoratore.

> **Important**
>
> Assicurati che la tua attività sia associata allo stesso AWS account della tua macchina a stati.

1. Nella console Step Functions, nel riquadro di navigazione a sinistra, scegli Attività.

2. ScegliereCreate activity (Crea attività).
3. Inserisci un nome per l'attività, ad esempio*get-greeting*, e quindi scegli Crea attività.
4. Quando viene creato il task dell'attività, prendere nota del relativo ARN, come illustrato nell'esempio seguente.

```
arn:aws:states:us-east-1:123456789012:activity:get-greeting
```

Passaggio 2: crea una macchina a stati

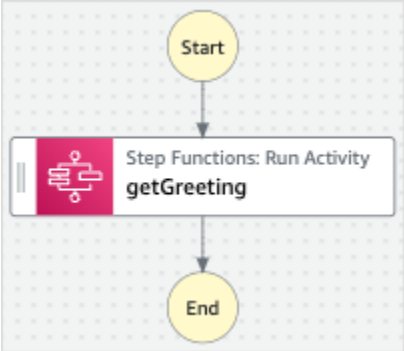
Crea una macchina a stati che determini quando l'attività viene richiamata e quando il lavoratore deve eseguire il lavoro principale, raccoglierne i risultati e restituirli. Per creare la macchina a stati, utilizzerai Workflow Studio. Editor del codice

1. Nella console Step Functions, nel riquadro di navigazione a sinistra, scegli Macchine a stati.
2. Nella pagina Macchine a stati, scegli Crea macchina a stati.
3. Nella finestra di dialogo Scegli un modello, seleziona Vuoto.
4. Scegli Seleziona. Questo apre Workflow Studio inModalità di progettazione.
5. Per questo tutorial, scriverai la definizione Amazon States Language (ASL) della tua macchina a stati nell'editor di codice. Per fare ciò, scegli Code.
6. Rimuovi il codice boilerplate esistente e incolla il codice seguente. Ricorda di sostituire l'ARN di esempio in questo codice con l'ARN dell'attività che hai creato in precedenza nel campo. `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
    }
  }
}
```

Questa è una descrizione della macchina a stati che utilizza il Amazon States Language (ASL). Definisce un singolo stato `Task` denominato `getGreeting`. Per ulteriori informazioni, consultare Struttura di una macchina a stati.

7. InRiquadro di visualizzazione del grafico, assicurati che il grafico del flusso di lavoro per la definizione ASL che hai aggiunto sia simile al grafico seguente.



8. Specificate un nome per la vostra macchina a stati. Per fare ciò, scegli l'icona di modifica accanto al nome della macchina a stati predefinita di MyStateMachine. Quindi, nella configurazione della macchina a stati, specifica un nome nella casella Nome macchina a stati.

   Per questo tutorial, inserisci il nome **`ActivityStateMachine`**.

9. (Facoltativo) Nella configurazione della macchina a stati, specificate altre impostazioni del flusso di lavoro, come il tipo di macchina a stati e il relativo ruolo di esecuzione.

   Per questo tutorial, mantieni tutte le selezioni predefinite nelle impostazioni della macchina a stati.

   Se in precedenza hai creato un ruolo IAM con le autorizzazioni corrette per la tua macchina a stati e desideri utilizzarlo, in Autorizzazioni seleziona Scegli un ruolo esistente, quindi seleziona un ruolo dall'elenco. Oppure seleziona Inserisci un ruolo ARN e quindi fornisci un ARN per quel ruolo IAM.

10. Nella finestra di dialogo Conferma la creazione del ruolo, scegli Conferma per continuare.

    Puoi anche scegliere Visualizza le impostazioni del ruolo per tornare alla configurazione della macchina a stati.

> **Note**
>
> Se elimini il ruolo IAM creato da Step Functions, Step Functions non può ricrearlo in un secondo momento. Allo stesso modo, se modifichi il ruolo (ad esempio, rimuovendo

Step Functions dai principi nella policy IAM), Step Functions non può ripristinare le impostazioni originali in un secondo momento.

Fase 3: implementazione di un lavoratore

Crea un lavoratore, ossia un programma responsabile di:

- Polling Step Functions per le attività che utilizzano l'azione `GetActivityTask` API.
- Esecuzione del lavoro dell'attività mediante codice, (ad esempio, il metodo `getGreeting()` nel codice seguente).
- Restituzione dei risultati mediante le operazioni API `SendTaskHeartbeat`, `SendTaskSuccess` e `SendTaskFailure`.

> **Note**
>
> Per un esempio più completo di un lavoratore di attività, consultare Esempio di lavoratore di attività in Ruby. Questo esempio fornisce un'implementazione basata su best practice, che può essere utilizzata come riferimento per il lavoratore di attività. Il codice implementa un modello consumatore-produttore con un numero configurabile di thread per poller e lavoratori di attività.

Per implementare il lavoratore

1. Crea un file denominato `GreeterActivities.java`.
2. Aggiungervi il seguente codice.

```
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**
>
> La classe `EnvironmentVariableCredentialsProvider` in questo esempio presuppone che le variabili di ambiente `AWS_ACCESS_KEY_ID` (o `AWS_ACCESS_KEY`) e `AWS_SECRET_KEY` (o `AWS_SECRET_ACCESS_KEY`) siano impostate. Per ulteriori informazioni su come fornire le credenziali richieste alla fabbrica, consulta AWSCredentialsProviderle sezioni AWS SDK for Java API Reference and Region for Development e Configurazione AWS delle credenziali e della regione per lo sviluppo nella AWS SDK for Java Developer Guide.
> Per impostazione predefinita, l' AWS SDK attenderà fino a 50 secondi per ricevere i dati dal server per qualsiasi operazione. `GetActivityTask` è un'operazione di polling lungo che attende per un massimo di 60 secondi la successiva attività disponibile. Per evitare la ricezione di un `SocketTimeoutException` errore, imposta SocketTimeout su 70 secondi.

3. Nell'elenco di parametri del costruttore `GetActivityTaskRequest().withActivityArn()`, sostituire il valore `ACTIVITY_ARN` con l'ARN del task di attività creato in precedenza.

Fase 4: Eseguire la macchina a stati

Quando si avvia l'esecuzione della macchina a stati, il lavoratore interroga Step Functions per le attività, esegue il suo lavoro (utilizzando l'input fornito) e ne restituisce i risultati.

1. Nella ***ActivityStateMachine***pagina, scegli Avvia esecuzione.

   Viene visualizzata la finestra di dialogo Avvia esecuzione.

2. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

   a. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

> **Note**
>
> Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

b. Nella casella Input, inserisci il seguente input JSON per eseguire il flusso di lavoro.

```
{
  "who": "AWS Step Functions"
}
```

c. Selezionare Start execution (Avvia esecuzione).

d. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

   Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Fase 5: esecuzione e arresto del lavoratore

Affinché il lavoratore esegua il polling di attività sulla macchina a stati, devi eseguire il lavoratore.

1. Dalla riga di comando, accedere alla directory in cui è stato creato `GreeterActivities.java`.
2. Per utilizzare l' AWS SDK, aggiungi il percorso completo delle `third-party` directory `lib` and alle dipendenze del tuo file di build e al tuo Java. `CLASSPATH` Per ulteriori informazioni, consulta Download ed estrazione dell'SDK nella Guida per gli sviluppatori.AWS SDK for Java
3. Compilare il file.

```
$ javac GreeterActivities.java
```

4. Esegui il file .

```
$ java GreeterActivities
```

5. Nella console Step Functions, vai alla pagina Dettagli di esecuzione.
6. Al termine dell'esecuzione, esamina i risultati dell'esecuzione.
7. Arrestare il lavoratore.

Iterare un ciclo con Lambda

In questo tutorial, implementi un modello di progettazione che utilizza una macchina a stati e una funzione AWS Lambda per eseguire l'iterazione di un loop un determinato numero di volte.

Utilizza questo modello di progettazione ogni volta che devi tenere traccia del numero di loop in una macchina a stati. Questa implementazione può consentirti di suddividere task di grandi dimensioni o esecuzioni di lungo durata in blocchi più piccoli oppure terminare un'esecuzione dopo un determinato numero di eventi. È possibile utilizzare un'implementazione simile per terminare e riavviare periodicamente un'esecuzione di lunga durata per evitare il superamento delle quote di servizio per AWS Step Functions o altri servizi. AWS Lambda AWS

Prima di iniziare, segui il Creazione di una macchina a stati Step Functions che utilizza Lambda tutorial per assicurarti di avere familiarità con l'uso congiunto di Lambda e Step Functions.

Passaggio 1: creare una funzione Lambda per iterare un conteggio

Utilizzando una funzione Lambda è possibile tenere traccia del numero di iterazioni di un ciclo nella macchina a stati. La seguente funzione Lambda riceve valori di input per `countindex`, e. `step` Restituisce quindi tali valori con un `index` aggiornato e un valore booleano denominato `continue`. La funzione Lambda viene impostata `continue` su `true` se `index` è minore di. `count`

La macchina a stati implementa quindi uno stato `Choice` che esegue alcune logiche di applicazione se `continue` è `true` o che termina se è `false`.

Creazione della funzione Lambda

1. Accedi alla console Lambda, quindi scegli Crea funzione.
2. Nella pagina Create function (Crea funzione), scegliere Author from scratch (Crea da zero).
3. Nella sezione Informazioni di base, configura la tua funzione Lambda come segue:

a. Nel campo Function name (Nome funzione), immettere `Iterator`.

b. In Runtime scegli Node.js.

c. In Modifica ruolo di esecuzione predefinito, scegli Crea un nuovo ruolo con autorizzazioni Lambda di base.

d. Scegli Crea funzione.

4. Copia il codice seguente per la funzione Lambda nel codice sorgente.

```
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
  })
}
```

Questo codice ammette valori di input per `count`, `index` e `step`. Incrementa l'`index` del valore di `step` e restituisce tali valori con il booleano `continue`. Il valore di `continue` è `true` se `index` è inferiore a `count`.

5. Seleziona Deploy (Implementa).

Fase 2: testare la funzione Lambda

Esegui la funzione Lambda con valori numerici per vederla in funzione. Puoi fornire valori di input per la tua funzione Lambda che imitano un'iterazione.

Per testare la tua funzione Lambda

1. Scegli Test (Esegui test).
2. Nella finestra di dialogo Configura evento di test, immettete `TestIterator` nella casella Nome evento.

3. Sostituire i dati di esempio con quanto segue.

```
{
  "Comment": "Test my Iterator function",
  "iterator": {
    "count": 10,
    "index": 5,
    "step": 1
  }
}
```

Tali valori simulano quelli generati da una macchina a stati durante un'iterazione. La funzione Lambda incrementa l'indice e restituisce `continue` quando `true` l'indice è inferiore a. `count` Per questo test, l'indice ha già subito un incremento a `5`. Il test verrà incrementato `6` e `index` impostato su. `continue` `true`

4. Scegli Crea.
5. Scegli Test per testare la tua funzione Lambda.

I risultati del test vengono visualizzati nella scheda Risultati di esecuzione.

6. Scegli la scheda Risultati di esecuzione per vedere l'output.

```
{
  "index": 6,
  "step": 1,
  "count": 10,
  "continue": true
}
```

> **Note**
>
> Se impostate su `index` 9 e testate nuovamente, gli `index` incrementi saranno pari a `10` e `continue` sarannofalse.

Fase 3: creazione di una macchina a stati

> Prima di lasciare la console Lambda...
>
> Copia la funzione Lambda ARN. Incollalo in una nota. Questo valore servirà nella fase successiva.

Successivamente, creerai una macchina a stati con i seguenti stati:

- `ConfigureCount`— Imposta i valori predefiniti per `count`index, `estep`.
- `Iterator`— Si riferisce alla funzione Lambda creata in precedenza, che trasmette i valori configurati in. `ConfigureCount`
- `IsCountReached`— Uno stato di scelta che continua il ciclo o procede allo `Done` stato, in base al valore restituito dalla `Iterator` funzione.
- `ExampleWork`— Un abbozzo per il lavoro che deve essere svolto. In questo esempio, il flusso di lavoro ha uno `Pass` stato, ma in una soluzione reale è probabile utilizzare un`Task`.
- `Done`— Stato finale del flusso di lavoro.

Per creare la macchina a stati nella console:

1. Apri la console Step Functions, quindi scegli Crea una macchina a stati.

   > **Important**
   >
   > La macchina a stati deve trovarsi nello stesso AWS account e nella stessa regione della funzione Lambda.

2. Seleziona il modello Blank.
3. Nel riquadro Codice, incolla il seguente codice JSON che definisce la macchina a stati.

   Per ulteriori informazioni su Amazon States Language, consulta State Machine Structure.

```
{
    "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. Sostituisci il `Iterator Resource` campo con l'ARN per la funzione `Iterator` Lambda che hai creato in precedenza.
5. Seleziona Config e inserisci un nome per la tua macchina a stati, ad esempio. *`IterateCount`*

> **Note**
>
> I nomi delle macchine a stati, delle esecuzioni e delle attività non devono superare gli 80 caratteri di lunghezza. Questi nomi devono essere univoci per l'account e la AWS regione e non devono contenere nessuno dei seguenti elementi:
>
> - Spazi bianchi
> - Caratteri jolly () `? *`
> - Caratteri tra parentesi quadre () `< > { } [ ]`
> - Caratteri speciali () `" # % \ ^ | ~ ` $ & , ; : /`
> - Caratteri di controllo (`\\u0000`- `\\u001f` o `\\u007f` -`\\u009f`).
>
> Se la macchina a stati è di tipo Express, è possibile fornire lo stesso nome a più esecuzioni della macchina a stati. Step Functions genera un ARN di esecuzione univoco per ogni esecuzione della macchina a stati Express, anche se più esecuzioni hanno lo stesso nome.
> Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

6. Per Tipo, accettate il valore predefinito di Standard. Per Autorizzazioni, scegli Crea nuovo ruolo.
7. Scegli Crea, quindi Conferma le creazioni del ruolo.

Fase 4: avvio di una nuova esecuzione

Una volta creata la macchina a stati, è possibile avviare un'esecuzione.

1. Nella IterateCountpagina, scegli Avvia esecuzione.

2. (Facoltativo) Per identificare l'esecuzione, puoi specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

   > **Note**
   >
   > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

3. Selezionare Start Execution (Avvia esecuzione).

   Viene avviata una nuova esecuzione della macchina a stati che mostra l'esecuzione in corso. Visualizzazione grafica della macchina a stati, che mostra lo stato di Iterator in blu per indicare lo stato in corso.

   L'esecuzione aumenta in fasi, tracciando il conteggio utilizzando la funzione Lambda. In ogni iterazione, esegue l'esempio di lavoro a cui si fa riferimento nello stato `ExampleWork` della macchina a stati.

   Quando il conteggio raggiunge il numero specificato nello stato `ConfigureCount` della macchina a stati, l'iterazione viene arrestata e l'esecuzione risulta completata. Visualizzazione grafica della macchina a stati, che mostra lo stato Iterator e lo stato Done in verde per indicare che entrambi hanno avuto successo.

Esecuzioni continue di flussi di lavoro a lungo termine come nuova esecuzione

AWS Step Functions è stato progettato per eseguire flussi di lavoro dalla durata e dal numero di fasi limitati. Le esecuzioni hanno una durata massima di un anno e un massimo di 25.000 eventi (vedi Quote).

Per le esecuzioni di lunga durata, per evitare di raggiungere la quota fissa di 25.000 voci nella cronologia degli eventi di esecuzione, si consiglia di avviare una nuova esecuzione del flusso di lavoro direttamente dallo `Task` stato di una macchina a stati. Ciò consente di suddividere i flussi di lavoro in macchine a stati più piccoli e di continuare il lavoro in corso in una nuova esecuzione. Per

avviare queste esecuzioni del flusso di lavoro, richiama l'azione `StartExecution` API dal tuo `Task` stato e passa i parametri necessari.

In alternativa, puoi anche implementare uno schema che utilizza una funzione Lambda per avviare una nuova esecuzione della tua macchina a stati per suddividere il lavoro in corso su più esecuzioni del flusso di lavoro.

Questo tutorial mostra entrambi gli approcci per continuare l'esecuzione del flusso di lavoro senza superare le quote di servizio.

Argomenti

- Utilizzo di un'azione API Step Functions per continuare una nuova esecuzione (consigliato)
- Utilizzo di una funzione Lambda per continuare una nuova esecuzione

Utilizzo di un'azione API Step Functions per continuare una nuova esecuzione (consigliato)

Step Functions può avviare l'esecuzione dei flussi di lavoro chiamando la propria API come servizio integrato. Si consiglia di utilizzare questo approccio per evitare il superamento delle quote di servizio per le esecuzioni di lunga durata.

Passaggio 1: creare una macchina a stati a esecuzione prolungata

Crea una macchina a stati a lunga durata che desideri avviare dallo `Task` stato di una macchina a stati diversa. Per questo tutorial, usa la macchina a stati che utilizza una funzione Lambda.

> **Note**
>
> Assicurati di copiare il nome e il nome Amazon Resource Name di questa macchina a stati in un file di testo per un uso successivo.

Passaggio 2: creare una macchina a stati per richiamare l'azione API Step Functions

Per avviare le esecuzioni dei flussi di lavoro da uno **`Task`** stato

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Nella finestra di dialogo Scegli un modello, seleziona Vuoto.

3. Scegli Seleziona. Questo apre Workflow Studio inModalità di progettazione.
4. Dalla scheda Azioni, trascina l'azione StartExecutionAPI e rilasciala nello stato vuoto denominato Drag first state here.
5. Scegli lo StartExecutionstato ed esegui le seguenti operazioni nella scheda Configurazione inModalità di progettazione:

   a. Rinomina lo stato in. **`Start nested execution`**

   b. Per Tipo di integrazione, scegli AWS SDK - new dall'elenco a discesa.

   c. In Parametri API, procedi come segue:

      i. Per esempio`StateMachineArn`, sostituisci l'Amazon Resource Name di esempio con l'ARN della tua macchina a stati. Ad esempio, inserisci l'ARN della macchina a stati che utilizza Lambda.

      ii. Per `Input` node, sostituisci il testo segnaposto esistente con il seguente valore:

      ```
      "Comment": "Starting workflow execution using a Step Functions API action"
      ```

      iii. Assicurati che i tuoi input in API Parameters siano simili ai seguenti:

      ```
      {
        "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. (Facoltativo) Scegliete Definizione nel ✓Inspector pannello per visualizzare la definizione generata automaticamente Amazon States Language (ASL) del flusso di lavoro.

   > **Tip**
   >
   > Puoi anche visualizzare la definizione ASL in Workflow Studio. Editor del codice Nell'editor di codice, puoi anche modificare la definizione ASL del tuo flusso di lavoro.

7. Specificate un nome per la vostra macchina a stati. Per fare ciò, scegli l'icona di modifica accanto al nome della macchina a stati predefinita di MyStateMachine. Quindi, nella configurazione della macchina a stati, specifica un nome nella casella Nome macchina a stati.

Per questo tutorial, inserisci il nome **`ParentStateMachine`**.

8. (Facoltativo) Nella configurazione della macchina a stati, specificate altre impostazioni del flusso di lavoro, come il tipo di macchina a stati e il relativo ruolo di esecuzione.

   Per questo tutorial, mantieni tutte le selezioni predefinite nelle impostazioni della macchina a stati.

   Se in precedenza hai creato un ruolo IAM con le autorizzazioni corrette per la tua macchina a stati e desideri utilizzarlo, in Autorizzazioni seleziona Scegli un ruolo esistente, quindi seleziona un ruolo dall'elenco. Oppure seleziona Inserisci un ruolo ARN e quindi fornisci un ARN per quel ruolo IAM.

9. Nella finestra di dialogo Conferma creazione del ruolo, scegli Conferma per continuare.

   Puoi anche scegliere Visualizza le impostazioni del ruolo per tornare alla configurazione della macchina a stati.

   > **Note**
   >
   > Se elimini il ruolo IAM creato da Step Functions, Step Functions non può ricrearlo in un secondo momento. Allo stesso modo, se modifichi il ruolo (ad esempio, rimuovendo Step Functions dai principi nella policy IAM), Step Functions non può ripristinare le impostazioni originali in un secondo momento.

Fase 3: Aggiornare la policy IAM

Per assicurarti che la tua macchina a stati disponga delle autorizzazioni per avviare l'esecuzione della macchina a stati che utilizza una funzione Lambda, devi allegare una policy in linea al ruolo IAM della tua macchina a stati. Per ulteriori informazioni, consulta Embedding Inline Policies nella IAM User Guide.

1. Nella ParentStateMachinepagina, scegli l'ARN del ruolo IAM per accedere alla pagina Ruoli IAM per la tua macchina a stati.
2. Assegna un'autorizzazione appropriata al ruolo IAM di ParentStateMachineper consentirgli di avviare l'esecuzione di un'altra macchina a stati. Per assegnare l'autorizzazione, procedi come segue:

a. Nella pagina IAM Roles, scegli Aggiungi autorizzazioni, quindi scegli Crea policy inline.

b. Nella pagina Crea policy, scegli la scheda JSON.

c. Sostituisci il testo esistente con la seguente politica.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "states:StartExecution"
          ],
          "Resource": [
              "arn:aws:states:us-
east-2:123456789012:stateMachine:LambdaStateMachine"
          ]
      }
  ]
}
```

d. Scegli Verifica policy.

e. Specificate un nome per la politica, quindi scegliete Crea politica.

Passaggio 4: Esegui la macchina a stati

Le esecuzioni di macchine a stati sono casi in cui si esegue il flusso di lavoro per eseguire attività.

1. Nella ParentStateMachinepagina, scegli Avvia esecuzione.

   Viene visualizzata la finestra di dialogo Avvia esecuzione.

2. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

   a. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

> **Note**
>
> Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

b. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

c. Selezionare Start execution (Avvia esecuzione).

d. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

   Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

3. Apri la LambdaStateMachinepagina e nota una nuova esecuzione attivata da. ParentStateMachine

Utilizzo di una funzione Lambda per continuare una nuova esecuzione

È possibile creare una macchina a stati che utilizza una funzione Lambda per avviare una nuova esecuzione prima del termine dell'esecuzione corrente. L'utilizzo di questo approccio per continuare il lavoro in corso con una nuova esecuzione consente di disporre di una macchina a stati in grado di suddividere lavori di grandi dimensioni in flussi di lavoro più piccoli o di avere una macchina a stati che funziona all'infinito.

Questo tutorial si basa sul concetto di utilizzo di una funzione Lambda esterna per modificare il flusso di lavoro, come dimostrato nel Iterare un ciclo con Lambda tutorial. Utilizzate la stessa funzione Lambda (`Iterator`) per iterare un ciclo per un numero specifico di volte. Inoltre, crei un'altra funzione Lambda per iniziare una nuova esecuzione del flusso di lavoro e per diminuire il conteggio

ogni volta che inizia una nuova esecuzione. La macchina a stati terminerà e riavvierà un'esecuzione per il numero di volte specificato nei dati di input.

La macchina a stati da creare implementa i seguenti stati.

Stato	Scopo
`ConfigureCount`	Uno `Pass` stato che configura i `step` valori `countindex`, e utilizzat i dalla funzione `Iterator` Lambda per eseguire le iterazioni del lavoro.
`Iterator`	Uno `Task` stato che fa riferimento alla funzione `Iterator` Lambda.
`IsCountReached`	Uno `Choice` stato che utilizza un valore booleano della `Iterator` funzione per decidere se la macchina a stati deve continuare a funzionare l'esempio o passare allo stato. `ShouldRestart`
`ExampleWork`	Uno `Pass` stato che rappresenta lo `Task` stato che eseguirebbe il lavoro in un'implementazione effettiva.
`ShouldRestart`	Uno `Choice` stato che utilizza il `executionCount` valore per decidere se terminare un'esecuzione e iniziarne un'altra, o semplicemente terminare.
`Restart`	Uno `Task` stato che utilizza una funzione Lambda per avviare una nuova esecuzione della macchina a stati. Anche questa funzione, come l'`Iterator`, decrementa i contatori. Lo `Restart` stato passa il valore decrementato del conteggio all'input della nuova esecuzion e.

Prerequisiti

Prima di iniziare, segui il Creazione di una macchina a stati Step Functions che utilizza Lambda tutorial per assicurarti di avere familiarità con l'uso congiunto di Lambda e Step Functions.

Argomenti

- Passaggio 1: creare una funzione Lambda per iterare un conteggio
- Fase 2: Creare una funzione Restart Lambda per avviare una nuova esecuzione di Step Functions

- Fase 3: Creare una macchina a stati
- Fase 4: Aggiornare la policy IAM
- Fase 5: Esegui la macchina a stati

Passaggio 1: creare una funzione Lambda per iterare un conteggio

> **Note**
>
> Se hai completato il Iterare un ciclo con Lambda tutorial, puoi saltare questo passaggio e utilizzare la funzione Lambda.

Questa sezione e il Iterare un ciclo con Lambda tutorial mostrano come utilizzare una funzione Lambda per tenere traccia di un conteggio, ad esempio il numero di iterazioni di un ciclo nella macchina a stati.

La seguente funzione Lambda riceve valori di input per `countindex`, e. `step` Restituisce quindi tali valori con un `index` aggiornato e un booleano denominato `continue`. La funzione Lambda viene impostata `continue` su `true` se `index` è minore di. `count`

La macchina a stati quindi implementa uno stato `Choice` che esegue alcune logiche di applicazione se `continue` è `true` o passa a `ShouldRestart` se `continue` è `false`.

Creare la funzione Iterate Lambda

1. Aprire la console Lambda, quindi scegliere Create function (Crea funzione).
2. Nella pagina Create function (Crea funzione), scegliere Author from scratch (Crea da zero).
3. Nella sezione Informazioni di base, configura la tua funzione Lambda come segue:

   a. Nel campo Function name (Nome funzione), immettere `Iterator`.

   b. Per Runtime scegli Node.js 16.x.

   c. Mantieni tutte le selezioni predefinite sulla pagina, quindi scegli Crea funzione.

   Quando viene creata la funzione Lambda, prendi nota del relativo Amazon Resource Name (ARN) nell'angolo in alto a destra della pagina, ad esempio:

   ```
   arn:aws:lambda:us-east-1:123456789012:function:Iterator
   ```

4. Copia il codice seguente per la funzione Lambda nella sezione Codice sorgente della pagina ***`Iterator`*** della console 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
  })
}
```

   Questo codice ammette valori di input per `count`, `index` e `step`. Incrementa l'`index` del valore di `step` e restituisce tali valori e il valore booleano di `continue`. Il valore di `continue` è `true` se `index` è inferiore a `count`.

5. Scegli Deploy per distribuire il codice.

Prova la funzione Iterate Lambda

Per apprezzarne il funzionamento, basta eseguire la funzione `Iterate` con valori numerici. Puoi fornire valori di input per la tua funzione Lambda che imitano un'iterazione per vedere quale output ottieni con valori di input specifici.

Per testare la tua funzione Lambda

1. Nella finestra di dialogo Configure test event (Configura evento test), selezionare Create new test event (Crea nuovo evento test), per poi specificare `TestIterator` come Event name (Nome evento).
2. Sostituire i dati di esempio con quanto segue.

```
{
  "Comment": "Test my Iterator function",
  "iterator": {
    "count": 10,
```

```
    "index": 5,
    "step": 1
  }
}
```

Tali valori simulano quelli generati da una macchina a stati durante un'iterazione. La funzione Lambda incrementa l'indice e restituisce as. `continue` `true` Quando raggiunge un valore pari o superiore a `count`, l'indice restituisce `continue` come `false`. Per questo test, l'indice ha già subito un incremento a `5`. I risultati dovrebbero annoverare l'incremento dell'`index` a `6` e l'impostazione di `continue` su `true`.

3. Scegli Crea.
4. Nella pagina ***`Iterator`*** della console Lambda, TestIteratorassicurati che sia elencato, quindi scegli Test.

   I risultati del test vengono visualizzati nella parte superiore della pagina. Selezionare Details (Dettagli) ed esaminare i risultati.

```
{
  "index": 6,
  "step": 1,
  "count": 10,
  "continue": true
}
```

> **Note**
>
> Un test eseguito con l'`index` impostato su `9` prevede l'incremento di tale `index` a `10` e l'impostazione di `continue` su `false`.

Fase 2: Creare una funzione Restart Lambda per avviare una nuova esecuzione di Step Functions

1. Aprire la console Lambda, quindi scegliere Create function (Crea funzione).
2. Nella pagina Create function (Crea funzione), scegliere Author from scratch (Crea da zero).
3. Nella sezione Informazioni di base, configura la tua funzione Lambda come segue:

   a. Nel campo Function name (Nome funzione), immettere `Restart`.

b. Per Runtime scegli Node.js 16.x.

4. Mantieni tutte le selezioni predefinite sulla pagina, quindi scegli Crea funzione.

   Quando viene creata la funzione Lambda, prendi nota del relativo Amazon Resource Name (ARN) nell'angolo in alto a destra della pagina, ad esempio:

   ```
   arn:aws:lambda:us-east-1:123456789012:function:Iterator
   ```

5. Copia il codice seguente per la funzione Lambda nella sezione Codice sorgente della pagina ***Riavvia della console*** Lambda.

   Il codice seguente riduce il conteggio del numero di esecuzioni e avvia una nuova esecuzione della macchina a stati, includendo il valore ridotto.

   ```
   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. Scegli Deploy per distribuire il codice.

Fase 3: Creare una macchina a stati

Ora che hai creato le tue due funzioni Lambda, crea una macchina a stati. In questa macchina a stati, gli stati `ShouldRestart` e `Restart` rispecchiano la suddivisione del lavoro in più esecuzioni.

Example ShouldRestart Stato della scelta

Il seguente estratto mostra lo stato. `ShouldRestart` `Choice` Questo stato determina se riavviare o meno l'esecuzione.

```
"ShouldRestart": {
"Type": "Choice",
"Choices": [
  {
    "Variable": "$.restart.executionCount",
    "NumericGreaterThan": 1,
    "Next": "Restart"
  }
],
```

Il valore `$.restart.executionCount` è incluso nell'input dell'esecuzione iniziale. Cala di uno a ogni chiamata della funzione `Restart` e poi viene incluso nell'input dell'esecuzione successiva.

Example Stato Task Restart

Il seguente estratto mostra lo stato. `Restart` `Task` Questo stato utilizza la funzione Lambda creata in precedenza per riavviare l'esecuzione e per diminuire il conteggio per tenere traccia del numero rimanente di esecuzioni da avviare.

```
"Restart": {
  "Type": "Task",
  "Resource": "arn:aws:lambda:us-east-1:123456789012:function:Restart",
  "Next": "Done"
},
```

Per creare la macchina a stati

1. Apri la console Step Functions e scegli Crea macchina a stati.

   > **Important**
   >
   > Assicurati che la tua macchina a stati utilizzi lo stesso AWS account e la stessa regione delle funzioni Lambda che hai creato in precedenza nei passaggi 1 e 2.

2. Nella finestra di dialogo Scegli un modello, seleziona Vuoto.
3. Scegli Seleziona. Questo apre Workflow Studio inModalità di progettazione.

4. Per questo tutorial, scriverai la definizione Amazon States Language (ASL) della tua macchina a stati in. Editor del codice Per fare ciò, scegli Codice.

5. Rimuovi il codice boilerplate esistente e incolla il codice seguente. Ricordati di sostituire gli ARN in questo codice con gli ARN delle funzioni Lambda che hai creato.

```
{
    "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. Specificate un nome per la vostra macchina a stati. Per fare ciò, scegli l'icona di modifica accanto al nome della macchina a stati predefinita di MyStateMachine. Quindi, nella configurazione della macchina a stati, specifica un nome nella casella Nome macchina a stati.

   Per questo tutorial, inserisci il nome **`ContinueAsNew`**.

7. (Facoltativo) Nella configurazione della macchina a stati, specificate altre impostazioni del flusso di lavoro, come il tipo di macchina a stati e il relativo ruolo di esecuzione.

   Per questo tutorial, mantieni tutte le selezioni predefinite nelle impostazioni della macchina a stati.

   Se in precedenza hai creato un ruolo IAM con le autorizzazioni corrette per la tua macchina a stati e desideri utilizzarlo, in Autorizzazioni seleziona Scegli un ruolo esistente, quindi seleziona un ruolo dall'elenco. Oppure seleziona Inserisci un ruolo ARN e quindi fornisci un ARN per quel ruolo IAM.

8. Nella finestra di dialogo Conferma la creazione del ruolo, scegli Conferma per continuare.

Puoi anche scegliere Visualizza le impostazioni del ruolo per tornare alla configurazione della macchina a stati.

> **Note**
>
> Se elimini il ruolo IAM creato da Step Functions, Step Functions non può ricrearlo in un secondo momento. Allo stesso modo, se modifichi il ruolo (ad esempio, rimuovendo Step Functions dai principi nella policy IAM), Step Functions non può ripristinare le impostazioni originali in un secondo momento.

9. Salva l'Amazon Resource Name (ARN) di questa macchina a stati in un file di testo. Dovrai fornire l'ARN mentre autorizzi la funzione Lambda per avviare una nuova esecuzione di Step Functions.

Fase 4: Aggiornare la policy IAM

Per assicurarti che la tua funzione Lambda disponga delle autorizzazioni necessarie per avviare una nuova esecuzione di Step Functions, allega una policy in linea al ruolo IAM che usi per la tua funzione Lambda. `Restart` Per ulteriori informazioni, consulta Embedding Inline Policies nella IAM User Guide.

> **Note**
>
> È possibile modificare la riga `Resource` dell'esempio precedente, affinché includa l'ARN della macchina a stati `ContinueAsNew`. Così facendo, si limita la policy in modo che vengano avviate solo esecuzioni di tale macchina a stati.

```
{
 "Version": "2012-10-17",
 "Statement": [
     {
         "Sid": "VisualEditor0",
         "Effect": "Allow",
         "Action": [
             "states:StartExecution"
         ],
         "Resource": "arn:aws:states:us-east-2:123456789012stateMachine:ContinueAsNew"
```

```
      }
  ]
}
```

Fase 5: Esegui la macchina a stati

Per avviare un'esecuzione, bisogna fornire un input che includa l'ARN della macchina a stati e l'`executionCount`, che rappresenta il numero di volte in cui è necessario avviare una nuova esecuzione.

1. Nella ContinueAsNewpagina, scegli Avvia esecuzione.

   Viene visualizzata la finestra di dialogo Avvia esecuzione.
2. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

   a. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

   > **Note**
   >
   > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

   b. Nella casella Input, inserisci il seguente input JSON per eseguire il flusso di lavoro.

   ```
   {
     "restart": {
       "StateMachineArn": "arn:aws:states:us-
   east-1:123456789012:stateMachine:ContinueAsNew",
       "executionCount": 4
     }
   }
   ```

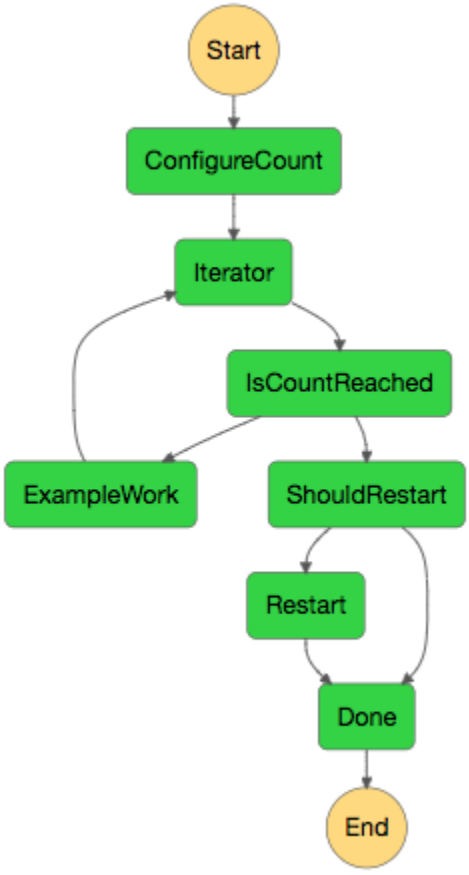
   c. Aggiornare il campo `StateMachineArn` con l'ARN della macchina a stati `ContinueAsNew`.

   d. Selezionare Start execution (Avvia esecuzione).

e. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

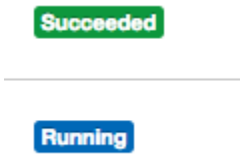
   Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

   La vista Grafico mostra la prima delle quattro esecuzioni. Prima del completamento, passando per lo stato `Restart` verrà avviata una nuova esecuzione.



   Una volta completata questa esecuzione, puoi esaminare l'esecuzione successiva in esecuzione. Seleziona il ContinueAsNewlink in alto per visualizzare l'elenco delle

esecuzioni. Dovresti vedere sia l'esecuzione chiusa di recente che l'esecuzione in corso avviata dalla funzione `Restart` Lambda.



Una volta completate, tutte le quattro esecuzioni saranno visibili nell'elenco. L'esecuzione avviata per prima mostra il nome scelto dall'utente; le altre, invece, presentano un nome generato.

Distribuire un progetto di approvazione umana di esempio

Questa esercitazione mostra come distribuire un progetto di approvazione umana che consente all'esecuzione AWS Step Functions di essere messa in pausa durante un'attività e di attendere che l'utente risponda a un'e-mail. Il flusso di lavoro passa allo stato successivo dopo che l'utente ha approvato l'attività per continuare.

L'implementazione dello AWS CloudFormation stack incluso in questo tutorial creerà tutte le risorse necessarie, tra cui:

- Risorse Amazon API Gateway
- E AWS Lambda funzioni
- Una macchina AWS Step Functions statale
- Un argomento e-mail di Amazon Simple Notification Service
- AWS Identity and Access Management Ruoli e autorizzazioni correlati

> **Note**
>
> Dovrai fornire un indirizzo email valido a cui avere accesso quando crei lo AWS CloudFormation stack.

Per ulteriori informazioni, consulta Lavorare con CloudFormation i modelli e la `AWS::StepFunctions::StateMachine` risorsa nella Guida per l'AWS CloudFormation utente.

Argomenti

- Fase 1: Creare un AWS CloudFormation modello
- Fase 2: Creare una pila
- Fase 3: approvare l'abbonamento Amazon SNS
- Passaggio 4: Esegui la macchina a stati
- AWS CloudFormation Codice sorgente del modello

Fase 1: Creare un AWS CloudFormation modello

1. Copiare il codice di esempio dalla sezione AWS CloudFormation Codice sorgente del modello.

2. Incolla il codice sorgente del AWS CloudFormation modello in un file sul tuo computer locale.

   Per questo esempio, il file è denominato `human-approval.yaml`.

Fase 2: Creare una pila

1. Accedi alla console AWS CloudFormation.
2. Scegli Crea pila, quindi scegli Con nuove risorse (standard).
3. Nella pagina Create stack (Crea stack), esegui le operazioni seguenti:

a. Nella sezione Prerequisito - Prepara modello, assicurati che sia selezionato Template is ready.

b. Nella sezione Specificare il modello, scegli Carica un file modello, quindi scegli Scegli file per caricare il `human-approval.yaml` file creato in precedenza che include il codice sorgente del modello.

4. Seleziona Successivo.

5. Nella pagina Specify stack details (Specifica dettagli), procedere come segue:

a. Per il nome dello stack, inserisci un nome per lo stack.

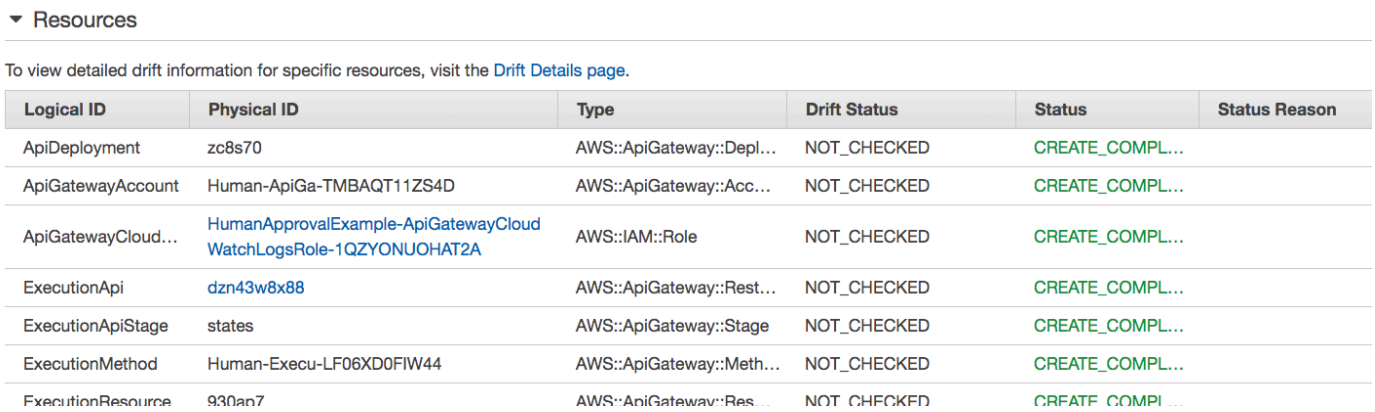
b. In Parametri, inserisci un indirizzo email valido. Utilizzerai questo indirizzo e-mail per iscriverti all'argomento Amazon SNS.

6. Scegli Avanti, quindi scegli nuovamente Avanti.

7. Nella pagina Revisione, scegli Riconosco che AWS CloudFormation potrebbe creare risorse IAM, quindi scegli Crea.

   AWS CloudFormation inizia a creare lo stack e visualizza lo stato CREATE_IN_PROGRESS. Quando il processo è completo, visualizza lo stato CREATE_COMPLETE. AWS CloudFormation

8. (Facoltativo) Per visualizzare le risorse nello stack, selezionare lo stack e scegliere la scheda Resources (Risorse).



Fase 3: approvare l'abbonamento Amazon SNS

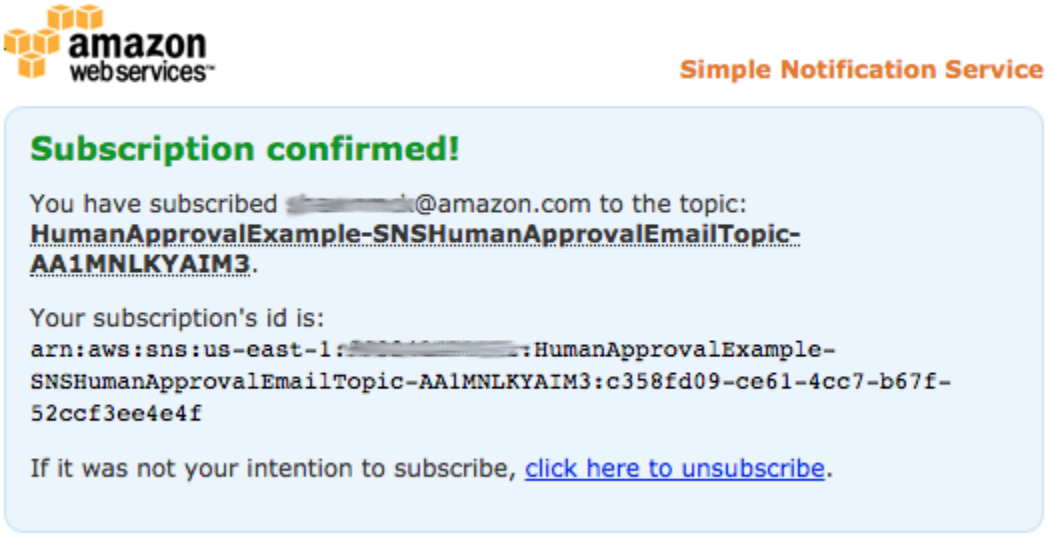
Una volta creato l'argomento Amazon SNS, riceverai un'e-mail di richiesta di conferma dell'iscrizione.

1. Apri l'account e-mail che hai fornito quando hai creato lo AWS CloudFormation stack.

2. Apri il messaggio AWS Notifica - Conferma dell'abbonamento da no-reply@sns.amazonaws.com

L'e-mail riporterà l'Amazon Resource Name per l'argomento Amazon SNS e un link di conferma.

3. Scegli il link conferma abbonamento.



Passaggio 4: Esegui la macchina a stati

1. Nella HumanApprovalLambdaStateMachinepagina, scegli Avvia esecuzione.

   Viene visualizzata la finestra di dialogo Avvia esecuzione.

2. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

   a. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

   > **Note**
   >
   > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

   b. Nella casella Input, inserisci il seguente input JSON per eseguire il flusso di lavoro.

```
{
```

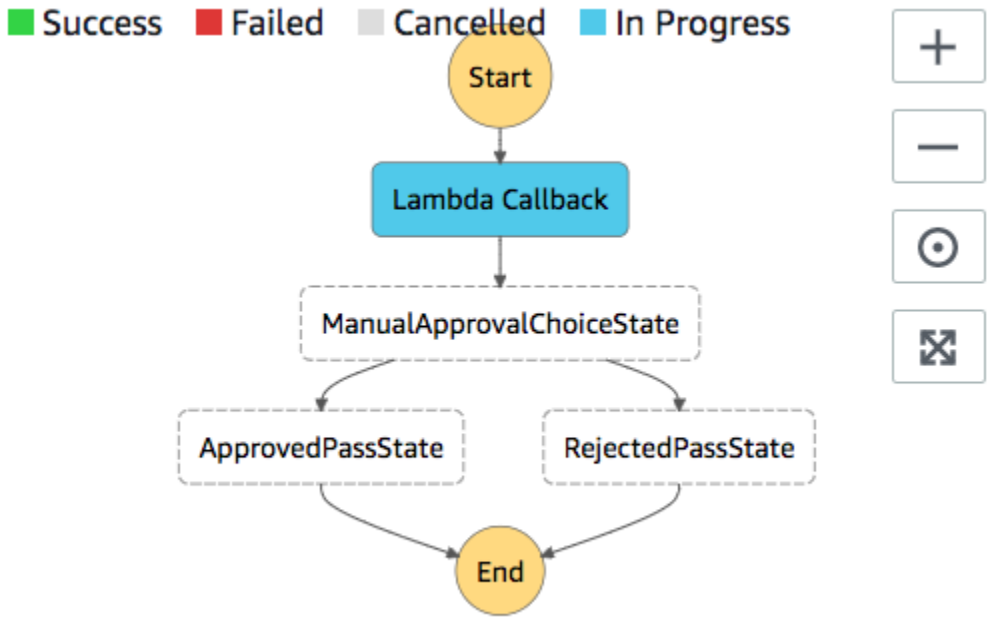
```
    "Comment": "Testing the human approval tutorial."
}
```

c. Selezionare Start execution (Avvia esecuzione).

L'esecuzione della macchina a ApprovalTeststati si avvia e si interrompe durante l'attività Lambda Callback.

d. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

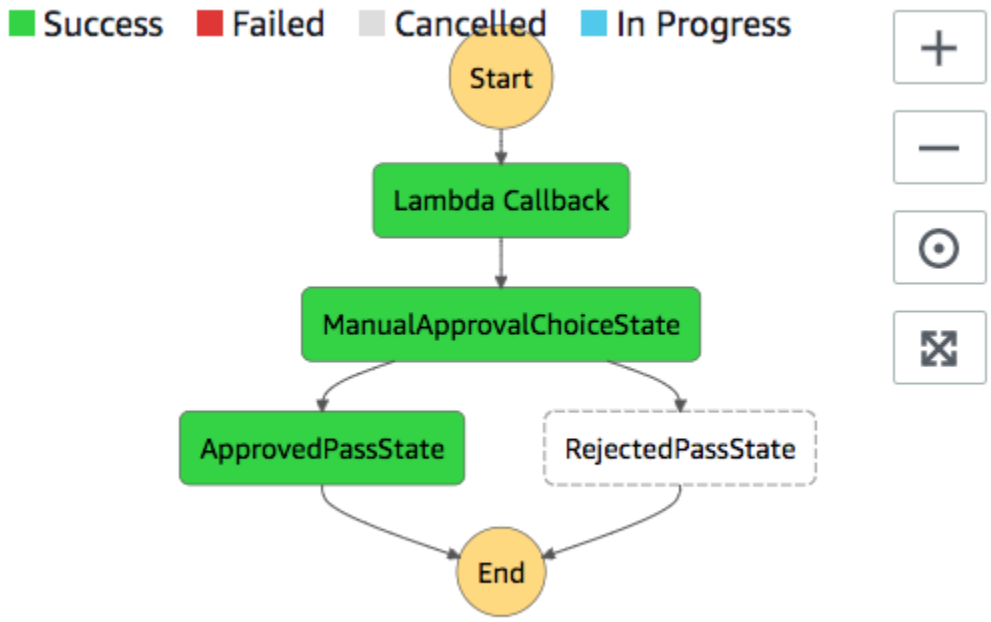


3. Nell'account e-mail che hai usato in precedenza per l'argomento Amazon SNS, apri il messaggio con l'oggetto da cui è richiesta l'approvazione. AWS Step Functions

Il messaggio include URL separati per Approva e Rifiuta.

4. Scegliere l'URL Approva.

Il flusso di lavoro prosegue in base alla tua scelta.



AWS CloudFormation Codice sorgente del modello

Utilizza questo AWS CloudFormation modello per implementare un esempio di flusso di lavoro del processo di approvazione umana.

```
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
```

Visualizza le tracce X-Ray in Step Functions

In questo tutorial, imparerai a usare X-Ray per tracciare gli errori che si verificano durante l'esecuzione di una macchina a stati. È possibile utilizzarlo AWS X-Rayper visualizzare i componenti della macchina a stati, identificare i punti deboli delle prestazioni e risolvere le richieste che hanno provocato un errore. In questo tutorial, creerai diverse funzioni Lambda che generano errori in modo casuale, che potrai quindi tracciare e analizzare utilizzando X-Ray.

Il Creazione di una macchina a stati Step Functions che utilizza Lambda tutorial ti guida nella creazione di una macchina a stati che chiama una funzione Lambda. Se hai completato il tutorial, vai al passaggio 2 e usa il ruolo AWS Identity and Access Management (IAM) che hai creato in precedenza.

Argomenti

- Fase 1: creare un ruolo IAM per Lambda
- Passaggio 2: creazione di una funzione Lambda
- Fase 3: Creare altre due funzioni Lambda
- Fase 4: Creare una macchina a stati
- Passaggio 5: Esegui la macchina a stati

Fase 1: creare un ruolo IAM per Lambda

Entrambi AWS Lambda AWS Step Functions possono eseguire codice e accedere a AWS risorse (ad esempio, dati archiviati nei bucket Amazon S3). Per mantenere la sicurezza, devi concedere a Lambda e Step Functions l'accesso a queste risorse.

Lambda richiede l'assegnazione di un ruolo AWS Identity and Access Management (IAM) quando si crea una funzione Lambda, allo stesso modo in cui Step Functions richiede l'assegnazione di un ruolo IAM quando si crea una macchina a stati.

Utilizzi la console IAM per creare un ruolo collegato al servizio.

Per creare un ruolo (console)

1. Accedi AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione della console IAM seleziona Ruoli. Quindi seleziona Create role (Crea ruolo).
3. Scegli il tipo AWS di ruolo Servizio, quindi scegli Lambda.
4. Scegli il caso d'uso Lambda. I casi d'uso sono definiti dal servizio in modo da includere la policy di attendibilità richiesta dal servizio. Quindi scegliere Next: Permissions (Successivo: Autorizzazioni).
5. Scegliere una o più policy di autorizzazione da collegare al ruolo (ad esempio, `AWSLambdaBasicExecutionRole`). Consultare Modello di autorizzazioni AWS Lambda.

   Selezionare la casella di controllo accanto alla policy che assegna le autorizzazioni da assegnare al ruolo, quindi scegliere Next: Review (Successivo: Rivedi).
6. Immettere un nome di ruolo in Role name (Nome ruolo).
7. (Facoltativo) In Role description (Descrizione ruolo) modifica la descrizione per il nuovo ruolo collegato ai servizi.
8. Verificare il ruolo e quindi scegliere Create role (Crea ruolo).

Passaggio 2: creazione di una funzione Lambda

La funzione Lambda genererà errori o timeout casuali, producendo dati di esempio da visualizzare in X-Ray.

> **Important**
>
> Assicurati che la tua funzione Lambda utilizzi lo stesso AWS account e la stessa AWS regione della tua macchina a stati.

1. Apri la console Lambda e scegli Crea funzione.
2. Nella sezione Create function (Crea funzione), scegliere Author from scratch (Crea da zero).
3. Nella sezione Informazioni di base, configura la tua funzione Lambda:
   a. Nel campo Function name (Nome funzione), immettere `TestFunction1`.
   b. Per Runtime, scegli Node.js 18.x.
   c. Per Role (Ruolo) selezionare Choose an existing role (Scegli un ruolo esistente).
   d. Per Ruolo esistente, seleziona il ruolo Lambda che hai creato in precedenza.

      > **Note**
      >
      > Se il ruolo IAM che hai creato non compare nell'elenco, il ruolo potrebbe richiedere ancora alcuni minuti per propagarsi a Lambda.

   e. Scegli Crea funzione.

      Quando viene creata la funzione Lambda, annota il relativo Amazon Resource Name (ARN) nell'angolo in alto a destra della pagina. Per esempio:

      ```
      arn:aws:lambda:us-east-1:123456789012:function:TestFunction1
      ```

4. Copia il codice seguente per la funzione Lambda nella sezione Codice funzione della pagina ***`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 }
};
```

Questo codice crea errori a tempo casuale, che verranno utilizzati per generare errori di esempio nella macchina a stati che possono essere visualizzati e analizzati utilizzando tracce a raggi X.

5. Selezionare Salva.

Fase 3: Creare altre due funzioni Lambda

Crea altre due funzioni Lambda.

1. Ripetere il passaggio 2 per creare altre due funzioni Lambda. Per la funzione successiva, in Nome funzione, immettere`TestFunction2`. Per l'ultima funzione, in Nome funzione, immettere`TestFunction3`.
2. Nella console Lambda, verifica che ora siano disponibili tre funzioni Lambda,, `TestFunction1` e. `TestFunction2` `TestFunction3`

Fase 4: Creare una macchina a stati

In questo passaggio, utilizzerai la console Step Functions per creare una macchina a stati con tre `Task` stati. Ogni `Task` stato farà riferimento a una delle tre funzioni Lambda.

1. Apri la console Step Functions e scegli Crea macchina a stati.

> **Important**
>
> Assicurati che la tua macchina a stati utilizzi lo stesso AWS account e la stessa regione delle funzioni Lambda che hai creato in precedenza nei passaggi 2 e 3.

2. Nella finestra di dialogo Scegli un modello, seleziona Vuoto.
3. Scegli Seleziona. Questo apre Workflow Studio inModalità di progettazione.
4. Per questo tutorial, scriverai la definizione Amazon States Language (ASL) della tua macchina a stati in. Editor del codice Per fare ciò, scegli Codice.

5. Rimuovi il codice boilerplate esistente e incolla il codice seguente. Nella definizione dello stato dell'attività, ricordati di sostituire gli ARN di esempio con gli ARN delle funzioni Lambda che hai creato.

```
{
  "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"
    }
  }
}
```

Questa è una descrizione della tua macchina a stati che utilizza Amazon States Language. Definisce tre `Task` stati denominati`CallTestFunction1`, `CallTestFunction2` e`CallTestFunction3`. Ciascuna richiama una delle tre funzioni Lambda. Per ulteriori informazioni, consultare Struttura di una macchina a stati.

6. Specificate un nome per la vostra macchina a stati. Per fare ciò, scegli l'icona di modifica accanto al nome della macchina a stati predefinita di MyStateMachine. Quindi, nella configurazione della macchina a stati, specifica un nome nella casella Nome macchina a stati.

   Per questo tutorial, inserisci il nome **`TraceFunctions`**.

7. (Facoltativo) Nella configurazione della macchina a stati, specificate altre impostazioni del flusso di lavoro, come il tipo di macchina a stati e il relativo ruolo di esecuzione.

   Per questo tutorial, in Configurazione aggiuntiva, scegli Abilita tracciamento X-Ray. Mantieni tutte le altre selezioni predefinite nelle impostazioni della macchina a stati.

   Se in precedenza hai creato un ruolo IAM con le autorizzazioni corrette per la tua macchina a stati e desideri utilizzarlo, in Autorizzazioni seleziona Scegli un ruolo esistente, quindi seleziona

un ruolo dall'elenco. Oppure seleziona Inserisci un ruolo ARN e quindi fornisci un ARN per quel ruolo IAM.

8. Nella finestra di dialogo Conferma la creazione del ruolo, scegli Conferma per continuare.

   Puoi anche scegliere Visualizza le impostazioni del ruolo per tornare alla configurazione della macchina a stati.

> **Note**
>
> Se elimini il ruolo IAM creato da Step Functions, Step Functions non può ricrearlo in un secondo momento. Allo stesso modo, se modifichi il ruolo (ad esempio, rimuovendo Step Functions dai principi nella policy IAM), Step Functions non può ripristinare le impostazioni originali in un secondo momento.

Passaggio 5: Esegui la macchina a stati

Le esecuzioni di macchine a stati sono casi in cui si esegue il flusso di lavoro per eseguire attività.

1. Nella ***`TraceFunctions`***pagina, scegli Avvia esecuzione.

   Viene visualizzata la pagina New execution (Nuova esecuzione).

2. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

   a. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

   > **Note**
   >
   > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

   b. Selezionare Start execution (Avvia esecuzione).

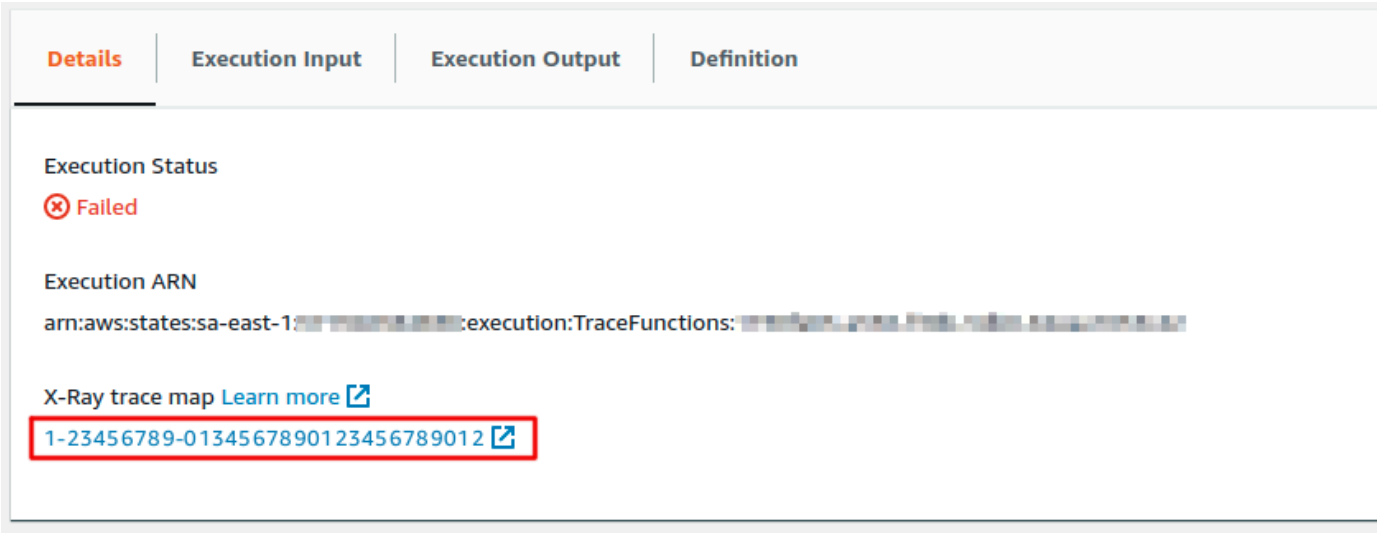
   c. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile

esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Esegui diverse (almeno tre) esecuzioni.

3. Una volta terminate le esecuzioni, segui il link alla mappa di tracciamento a raggi X. È possibile visualizzare la traccia mentre un'esecuzione è ancora in esecuzione, ma è consigliabile visualizzare i risultati dell'esecuzione prima di visualizzare la mappa di traccia a raggi X.



4. Visualizza la mappa dei servizi per identificare dove si verificano gli errori, le connessioni con latenza elevata o le tracce delle richieste che non hanno avuto esito positivo. In questo esempio, puoi vedere quanto traffico riceve ciascuna funzione. `TestFunction2`è stato chiamato più spesso di`TestFunction3`, ed `TestFunction1` è stato chiamato più del doppio di`TestFunction2`.

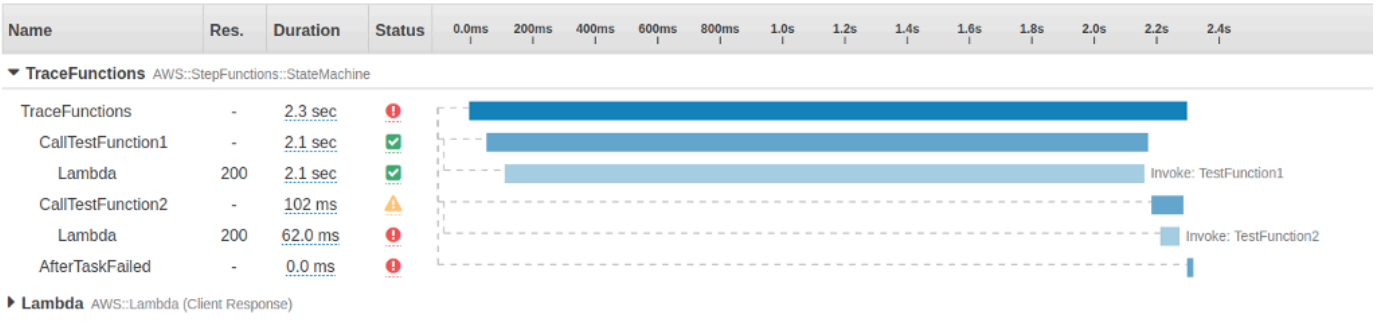
   La mappa del servizio indica lo stato di salute di ogni nodo colorandolo in base al rapporto delle chiamate concluse con successo rispetto a errori e malfunzionamenti:

   - Verde per le chiamate concluse con successo
   - Rosso per i malfunzionamenti del server (errori serie 500)
   - Giallo per gli errori client (errori serie 400)

- Viola per gli errori di superamento dei limiti (429 Too Many Requests)

Puoi anche scegliere un nodo di servizio per visualizzare le richieste per quel nodo o un edge tra due nodi per visualizzare le richieste che hanno percorso quella connessione.

5. Visualizza la mappa di tracciamento a raggi X per vedere cosa è successo per ogni esecuzione. La visualizzazione della sequenza temporale mostra una gerarchia di segmenti e sottosegmenti. La prima voce nell'elenco è il segmento, che rappresenta tutti i dati registrati tramite il servizio per una singola richiesta. Sotto il segmento sono presenti i sottosegmenti. Questo esempio mostra i sottosegmenti registrati dalle funzioni Lambda.



Per ulteriori informazioni sulla comprensione delle tracce a raggi X e sull'utilizzo di X-Ray with Step Functions, vedere AWS X-Ray e Step Functions

Raccogli informazioni sui bucket Amazon S3 utilizzando AWS le integrazioni dei servizi SDK

Questo tutorial mostra come eseguire un'integrazione AWS SDK con Amazon Simple Storage Service. La macchina a stati che crei in questo tutorial raccoglie informazioni sui tuoi bucket Amazon S3, quindi elenca i bucket insieme alle informazioni sulla versione per ogni bucket nella regione corrente.

Argomenti

- Fase 1: Creare la macchina a stati
- Passaggio 2: aggiungi le autorizzazioni necessarie per il ruolo IAM
- Passaggio 3: Esegui l'esecuzione di una macchina a stati standard
- Fase 4: Eseguire l'esecuzione di una macchina a stati Express

Fase 1: Creare la macchina a stati

Utilizzando la console Step Functions, creerai una macchina a stati che include uno `Task` stato per elencare tutti i bucket Amazon S3 nell'account e nella regione correnti. Quindi, aggiungerai un altro `Task` stato che richiama l'`HeadBucket`API per verificare se il bucket restituito è accessibile nella regione corrente. Se il bucket non è accessibile, la chiamata `HeadBucket` API restituisce l'errore. `S3.S3Exception` Includerai un `Catch` blocco per catturare questa eccezione e uno `Pass` stato come stato di fallback.

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Nella finestra di dialogo Scegli un modello, seleziona Vuoto.
3. Scegli Seleziona. Questo apre Workflow Studio inModalità di progettazione.
4. Per questo tutorial, scriverai la definizione Amazon States Language (ASL) della tua macchina a stati in. Editor del codice Per fare ciò, scegli Code.
5. Rimuovi il codice boilerplate esistente e incolla la seguente definizione di macchina a stati.

```
{
  "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. Specificate un nome per la vostra macchina a stati. Per fare ciò, scegli l'icona di modifica accanto al nome della macchina a stati predefinita di MyStateMachine. Quindi, nella configurazione della macchina a stati, specifica un nome nella casella Nome macchina a stati.

   Per questo tutorial, inserisci il nome **`Gather-S3-Bucket-Info-Standard`**.

7. (Facoltativo) Nella configurazione della macchina a stati, specificate altre impostazioni del flusso di lavoro, come il tipo di macchina a stati e il relativo ruolo di esecuzione.

   Conserva tutte le selezioni predefinite nelle impostazioni della macchina a stati.

   Se in precedenza hai creato un ruolo IAM con le autorizzazioni corrette per la tua macchina a stati e desideri utilizzarlo, in Autorizzazioni seleziona Scegli un ruolo esistente, quindi seleziona un ruolo dall'elenco. Oppure seleziona Inserisci un ruolo ARN e quindi fornisci un ARN per quel ruolo IAM.

8. Nella finestra di dialogo Conferma la creazione del ruolo, scegli Conferma per continuare.

   Puoi anche scegliere Visualizza le impostazioni del ruolo per tornare alla configurazione della macchina a stati.

   > **Note**
   >
   > Se elimini il ruolo IAM creato da Step Functions, Step Functions non può ricrearlo in un secondo momento. Allo stesso modo, se modifichi il ruolo (ad esempio, rimuovendo Step Functions dai principi nella policy IAM), Step Functions non può ripristinare le impostazioni originali in un secondo momento.

   Nel passaggio 2, aggiungerai le autorizzazioni mancanti al ruolo della macchina a stati.

Passaggio 2: aggiungi le autorizzazioni necessarie per il ruolo IAM

Per raccogliere informazioni sui bucket Amazon S3 nella tua regione attuale, devi fornire alla tua macchina a stati le autorizzazioni necessarie per accedere ai bucket Amazon S3.

1. Nella pagina della macchina a stati, scegli ARN del ruolo IAM per aprire la pagina Ruoli per il ruolo della macchina a stati.
2. Scegli Aggiungi autorizzazioni, quindi seleziona Crea policy inline.
3. Scegli la scheda JSON, quindi incolla le seguenti autorizzazioni nell'editor JSON.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:GetBucketVersioning"
            ],
            "Resource": "*"
        }
    ]
}
```

4. Scegli Verifica policy.
5. In Rivedi policy, per Nome della policy inserisci **`s3-bucket-permissions`**.
6. Scegli Crea policy.

Passaggio 3: Esegui l'esecuzione di una macchina a stati standard

1. Nella pagina Gather-S3-Bucket-Info-Standard, scegli Avvia esecuzione.
2. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

    a. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

> **Note**
>
> Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

b. Selezionare Start execution (Avvia esecuzione).

c. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

   Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Fase 4: Eseguire l'esecuzione di una macchina a stati Express

1. Creare una macchina a stati Express utilizzando la definizione della macchina a stati fornita nel passaggio 1. Assicurati di includere anche le autorizzazioni necessarie per i ruoli IAM, come spiegato nella Fase 2.

   > **Tip**
   >
   > Per distinguerla dalla macchina Standard che hai creato in precedenza, assegna alla macchina a stati Express il nome. **`Gather-S3-Bucket-Info-Express`**

2. Nella pagina Gather-S3-Bucket-Info-Standard, scegliete Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

   a. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

> **Note**
>
> Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

b. Selezionare Start execution (Avvia esecuzione).

c. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

   Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Strumenti per sviluppatori

Le seguenti risorse contengono informazioni aggiuntive sulla creazione di flussi di lavoro serverless e sull'utilizzo di macchine a stati:

- AWS CDK

- AWS Toolkit per VS Code

Gli argomenti seguenti contengono informazioni che insegnano a creare, testare ed eseguire il debug di macchine a stati.

Argomenti

- Opzioni di sviluppo
- AWS Step Functions e AWS SAM
- Utilizzo di Workflow Studio in Application Composer
- Creazione di una macchina a stati Lambda per Step Functions utilizzando AWS CloudFormation
- Creazione di una macchina a Lambda stati per Step Functions l'utilizzo AWS CDK
- Creazione di un'API Gateway REST con Synchronous Express State Machine utilizzando AWS CDK
- AWS Step Functions SDK per Data Science per Python
- Distribuzione di macchine a stati utilizzando Terraform

Opzioni di sviluppo

Puoi implementare le tue macchine a AWS Step Functions stati in diversi modi, ad esempio utilizzando la console, gli SDK o una versione locale di Step Functions per il test e lo sviluppo.

Argomenti

- Console Step Functions
- AWS SDK
- Flussi di lavoro Standard ed Express
- API del servizio HTTPS

- Ambienti di sviluppo
- Endpoints
- AWS CLI
- Step Functions Local
- AWS Toolkit for Visual Studio Code
- AWS Serverless Application Model e Step Functions
- Terraform e Step Functions
- Supporto per il formato di definizione

Console Step Functions

È possibile definire una macchina a stati utilizzando la console Step Functions. Puoi scrivere macchine a stati complesse nel cloud senza utilizzare un ambiente di sviluppo locale utilizzando AWS Lambda to per fornire codice per le tue attività. Una volta scritto, puoi quindi utilizzare la console Step Functions per definire la tua macchina a stati utilizzando Amazon States Language.

Il tutorial Creating a Lambda State Machine utilizza questa tecnica per creare una semplice macchina a stati, eseguirla e visualizzarne i risultati.

Simulatore di flusso di dati

Puoi progettare, implementare ed eseguire il debug dei flussi di lavoro nella console Step Functions. Puoi anche controllare il flusso di dati nei flussi di lavoro utilizzando l'elaborazione di `JsonPath` input e output. Usa il simulatore di flusso di dati nella console Step Functions per scoprire come le informazioni fluiscono da uno stato all'altro e per capire come filtrare e manipolare i dati. Questo strumento simula ciascuno dei campi utilizzati da Step Functions per elaborare i dati, ad esempio `InputPathParameters`, `ResultSelectorOutputPath`, e`ResultPath`.

Per ulteriori informazioni, consulta Simulatore di flusso di dati.

AWS SDK

Step Functions è supportato dagli AWS SDK per Java, .NET, Ruby, PHP, Python (Boto 3), JavaScript Go e C++. Questi SDK offrono un modo pratico per utilizzare le azioni dell'API HTTPS di Step Functions in più linguaggi di programmazione.

Puoi sviluppare macchine a stati, attività o starter di macchine a stati utilizzando le operazioni API esposte da queste librerie SDK, Puoi anche accedere alle operazioni di visibilità utilizzando queste librerie per sviluppare i tuoi strumenti di monitoraggio e reporting Step Functions.

Per utilizzare Step Functions con altri AWS servizi, consulta la documentazione di riferimento per gli AWS SDK e gli strumenti attuali per Amazon Web Services.

> **Note**
>
> Step Functions supporta solo gli endpoint HTTPS.

Flussi di lavoro Standard ed Express

Quando si crea una nuova macchina a stati, è necessario selezionare un Type Standard o Express. In entrambi i casi, definisci la tua macchina a stati utilizzando Amazon States Language. Le esecuzioni della macchina a stati si comportano in modo diverso, a seconda di cosa si seleziona per Type (Tipo). Il tipo scelto non può essere modificato dopo la creazione della macchina a stati.

Per ulteriori informazioni, consulta Registrazione utilizzandoCloudWatchRegistri.

API del servizio HTTPS

Step Functions fornisce operazioni di servizio accessibili tramite richieste HTTPS. È possibile utilizzare queste operazioni per comunicare direttamente con Step Functions e sviluppare librerie personalizzate in qualsiasi linguaggio in grado di comunicare con Step Functions tramite HTTPS.

Puoi sviluppare macchine a stati, lavoratori o starter di macchine a stati utilizzando le operazioni API di servizio. Puoi inoltre accedere a operazioni di visibilità mediante operazioni API per sviluppare strumenti di monitoraggio e reporting personalizzati.

Per informazioni dettagliate sulle azioni API, consulta l'AWS Step Functions API Reference.

Ambienti di sviluppo

È necessario configurare un ambiente di sviluppo compatibile con il linguaggio di programmazione che si intende utilizzare.

Ad esempio, per sviluppare per Step Functions utilizzando Java, è necessario installare un ambiente di sviluppo Java, come il AWS SDK for Java, su ciascuna delle workstation di sviluppo. Se utilizzi

l'IDE di Eclipse per gli sviluppatori Java, devi anche installare il AWS Toolkit for Eclipse. Questo plugin Eclipse aggiunge caratteristiche utili per lo sviluppo su AWS.

Se il linguaggio di programmazione richiede un ambiente di runtime, è necessario configurare l'ambiente su ogni computer in cui verranno eseguiti questi processi.

Endpoints

Per ridurre la latenza e archiviare i dati in una posizione che soddisfi i requisiti, Step Functions fornisce endpoint in diverse AWS regioni.

Ogni endpoint in Step Functions è completamente indipendente. Una macchina a stati o attività esiste solo all'interno della regione in cui è stata creata. Qualsiasi macchina a stati e attività che crei in una regione non condivide dati o attributi con quelle create in un'altra regione. Ad esempio, è possibile registrare una macchina a stati denominata `STATES-Flows-1` in due regioni diverse. La macchina a `STATES-Flows-1` stati in una regione non condividerà dati o attributi con la macchina a `STATES-Flow-1` stati dell'altra regione.

Per un elenco degli endpoint di Step Functions, consulta AWS Step Functions Regions and Endpoints in. Riferimenti generali di AWS

AWS CLI

È possibile accedere a molte funzionalità di Step Functions da AWS Command Line Interface (AWS CLI). AWS CLI È un'alternativa all'utilizzo della console Step Functions o, in alcuni casi, alla programmazione utilizzando le azioni dell'API Step Functions. Ad esempio, è possibile utilizzare il AWS CLI per creare una macchina a stati e quindi elencare le macchine a stati esistenti.

È possibile utilizzare i comandi Step Functions in AWS CLI per avviare e gestire le esecuzioni, eseguire sondaggi sulle attività, registrare i battiti cardiaci delle attività e altro ancora. Per un elenco completo dei comandi Step Functions, le descrizioni degli argomenti disponibili e gli esempi che ne mostrano l'utilizzo, consulta il AWS CLI Command Reference.

AWS CLI i comandi seguono da vicino l'Amazon States Language, quindi puoi utilizzarli AWS CLI per conoscere le azioni dell'API Step Functions. Puoi anche utilizzare le tue conoscenze esistenti in materia di API per prototipare codice o eseguire azioni Step Functions dalla riga di comando.

Step Functions Local

Per scopi di test e sviluppo, puoi installare ed eseguire Step Functions sul tuo computer locale. Con Step Functions Local, puoi avviare un'esecuzione su qualsiasi macchina.

La versione locale di Step Functions può richiamare AWS Lambda funzioni, sia in AWS esecuzione locale che durante l'esecuzione. È inoltre possibile coordinare altri AWS servizi supportati. Per ulteriori informazioni, consulta Test di macchine a stato locale.

> **Note**
>
> Step Functions Local utilizza account fittizi per funzionare.

AWS Toolkit for Visual Studio Code

È possibile utilizzare VS Code per interagire con macchine a stati remoti e sviluppare macchine a stati localmente. È possibile creare o aggiornare macchine a stati, elencare le macchine a stati esistenti ed eseguire o scaricare una macchina a stati. VS Code consente inoltre di creare nuove macchine a stati da modelli, osservare una visualizzazione della macchina a stati e fornire frammenti, completamento e convalida del codice.

Per ulteriori informazioni, consulta la Guida AWS Toolkit for Visual Studio Code per l'utente

AWS Serverless Application Model e Step Functions

Step Functions è integrato con AWS Serverless Application Model, che consente di integrare i flussi di lavoro con funzioni, API ed eventi Lambda per creare applicazioni serverless.

Puoi anche utilizzare la AWS SAM CLI insieme a AWS Toolkit for Visual Studio Code come parte di un'esperienza integrata.

Per ulteriori informazioni, consultare AWS Step Functions e AWS SAM.

Terraform e Step Functions

Terraform by HashiCorp è un framework per la creazione di applicazioni che utilizzano l'infrastruttura come codice (IaC). Con Terraform, puoi creare macchine a stati e utilizzare funzionalità, come l'anteprima delle implementazioni dell'infrastruttura e la creazione di modelli riutilizzabili. I modelli Terraform ti aiutano a mantenere e riutilizzare il codice suddividendolo in blocchi più piccoli.

Per ulteriori informazioni, consulta Distribuzione di macchine a stati utilizzando Terraform.

Supporto per il formato di definizione

Step Functions offre una varietà di strumenti che consentono di fornire le definizioni delle macchine a stati in diversi formati. Una definizione di Amazon States Language (ASL) che specifica i dettagli della tua macchina a stati può essere fornita come stringa o come oggetto serializzato utilizzando JSON o YAML.

> **Note**
>
> YAML consente commenti a riga singola. Qualsiasi commento YAML fornito nella parte relativa alla definizione della macchina a stati di un modello non verrà inserito nella definizione della risorsa creata. È invece possibile utilizzare la `Comment` proprietà all'interno della definizione della macchina a stati. Per ulteriori informazioni, consulta la Struttura della macchina a stati pagina.

La tabella seguente mostra quali strumenti supportano le definizioni basate su ASL.

Supporto del formato di definizione per strumento

	JSON	YAML	Lingua degli Stati Amazon Stringificata		
Console Step Functions	✓				
API del servizio HTTPS			✓		
AWS CLI			✓		
Step Functions Local			✓		

	JSON	YAML	Lingua degli Stati Amazon Stringificata		
AWS Toolkit for Visual Studio Code	✓	✓			
AWS SAM	✓	✓			
AWS CloudFormation	✓	✓	✓		

> **Note**
>
> AWS CloudFormation e consentono AWS SAM inoltre di caricare le definizioni delle macchine a stati su Amazon S3 in formato JSON o YAML e di fornire la posizione Amazon S3 della definizione nel modello. Ciò può migliorare la leggibilità dei modelli quando la definizione della macchina a stati è complessa. Per ulteriori informazioni, consulta la pagina AWS::StepFunctions::StateMachine S3Location.

I seguenti AWS CloudFormation modelli di esempio mostrano come è possibile fornire la stessa definizione di macchina a stati utilizzando formati di input diversi.

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 e AWS SAM

Puoi utilizzare la AWS SAM CLI insieme a AWS Toolkit for Visual Studio Code come parte di un'esperienza integrata per creare macchine a stati localmente. Puoi creare un'applicazione serverless con AWS SAM, quindi creare la tua macchina a stati nell'IDE VS Code. Quindi puoi convalidare, impacchettare e distribuire le tue risorse. Facoltativamente, puoi anche pubblicare su. AWS Serverless Application Repository

> **Tip**
>
> Per distribuire un'applicazione serverless di esempio che avvia un flusso di lavoro Step Functions utilizzando AWS SAM to your Account AWS, vedi Modulo 11 - Deploy with AWS SAM of The Workshop. AWS Step Functions

Argomenti

- Perché usare Step Functions con AWS SAM?
- Integrazione di Step Functions con le AWS SAM specifiche
- Integrazione delle funzioni a fasi con l’interfaccia a riga di comando SAM
- DefinitionSubstitutions nei modelli AWS SAM
- Passaggi successivi

Perché usare Step Functions con AWS SAM?

Quando usi Step Functions con AWS SAM puoi:

- Inizia a utilizzare un modello AWS SAM di esempio.
- Costruire la tua macchina a stati nella tua applicazione serverless.
- Utilizza la sostituzione delle variabili per sostituire gli ARN nella tua macchina a stati al momento della distribuzione.

  AWS CloudFormationsupporti `DefinitionSubstitutions`che consentono di aggiungere riferimenti dinamici nella definizione del flusso di lavoro a un valore fornito nel modello. CloudFormation È possibile aggiungere riferimenti dinamici aggiungendo sostituzioni alla definizione del flusso di lavoro utilizzando la `${dollar_sign_brace}` notazione. È inoltre necessario definire questi riferimenti dinamici nella `DefinitionSubstitutions` proprietà della StateMachine risorsa nel modello. CloudFormation Queste sostituzioni vengono sostituite con valori effettivi durante il processo di creazione dello CloudFormation stack. Per ulteriori informazioni, consulta DefinitionSubstitutions nei modelli AWS SAM.
- Specificate il ruolo della vostra macchina a stati utilizzando AWS SAM i modelli di policy.
- Avvia esecuzioni di macchine a stati con API Gateway, EventBridge eventi o in base a una pianificazione all'interno del tuo AWS SAM modello.

Integrazione di Step Functions con le AWS SAM specifiche

È possibile utilizzare i modelli di AWS SAM policy per aggiungere autorizzazioni alla macchina a stati. Con queste autorizzazioni, puoi orchestrare le funzioni Lambda e AWS altre risorse per formare flussi di lavoro complessi e solidi.

Integrazione delle funzioni a fasi con l’interfaccia a riga di comando SAM

Step Functions è integrato con la AWS SAM CLI. Utilizzare questa opzione per sviluppare rapidamente una macchina a stati nell'applicazione serverless.

Prova il Creazione di una macchina a stati Step FunctionsAWS SAM tutorial per imparare a usare per creare macchine AWS SAM a stati.

Le funzioni AWS SAM CLI supportate includono:

Comando CLI	Descrizione
sam init	Inizializza un'applicazione serverless con un modello. AWS SAM Può essere utilizzata con un modello SAM per le funzioni Step.
sam validate	Convalida un modello. AWS SAM
sam package	Pacchettizza un' AWS SAM applicazione. Crea un file ZIP del codice e delle dipendenze, quindi lo carica su Amazon S3. Restituisce quindi una copia del modello AWS SAM , sostituendo i riferimenti agli artefatti locali con la posizione Amazon S3 in cui il comando ha caricato gli artefatti.
sam deploy	Distribuisce un'applicazione. AWS SAM
sam publish	Pubblica un' AWS SAM applicazione su. AWS Serverless Application Repository Questo comando accetta un AWS SAM modello impacchettato e pubblica l'applicazione nella regione specificata.

> **Note**
>
> Quando si utilizza AWS SAM local, è possibile emulare Lambda e API Gateway localmente. Tuttavia, non è possibile emulare Step Functions localmente utilizzando AWS SAM.

DefinitionSubstitutions nei modelli AWS SAM

È possibile definire macchine a stati utilizzando CloudFormation modelli conAWS SAM. Con AWS SAM, è possibile definire la macchina a stati in linea nel modello o in un file separato. Il AWS SAM modello seguente include una macchina a stati che simula un flusso di lavoro di compravendita di azioni. Questa macchina a stati richiama tre Lambda funzioni per controllare il prezzo di un'azione e determinare se acquistare o vendere l'azione. Questa transazione viene quindi registrata in una

Amazon DynamoDB tabella. Gli ARN per le Lambda funzioni e la DynamoDB tabella nel modello seguente vengono specificati utilizzando `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
```

Il codice seguente è la definizione della macchina a stati contenuta nel file `stock_trader.asl.json` utilizzato nel Creazione di una macchina a stati Step FunctionsAWS SAM tutorial. Questa definizione di macchina a stati ne contiene diverse `DefinitionSubstitutions` denotate dalla notazione. `${dollar_sign_brace}` Ad esempio, invece di specificare una Lambda funzione statica ARN per `Check Stock Value` l'attività, viene utilizzata la `${StockCheckerFunctionArn}` sostituzione. Questa sostituzione è definita nella proprietà del modello. DefinitionSubstitutions `DefinitionSubstitutions`è una mappa di coppie chiave-valore per la risorsa della macchina a stati. In`DefinitionSubstitutions`, ${StockCheckerFunctionArn} esegue il mapping all'ARN della `StockCheckerFunction` risorsa utilizzando la funzione CloudFormation intrinseca. `!GetAtt` Quando si distribuisce il AWS SAM modello, i valori contenuti `DefinitionSubstitutions` nel modello vengono sostituiti con i valori effettivi.

```
{
    "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
        }
    }
}
```

Passaggi successivi

Puoi saperne di più sull'utilizzo di Step Functions AWS SAM con le seguenti risorse:

- Completa il Creazione di una macchina a stati Step FunctionsAWS SAM tutorial per creare una macchina a stati con AWS SAM.
- Specificate una AWS::Serverless::StateMachinerisorsa.
- Trova modelli di criteri AWS SAM da utilizzare.
- Utilizzare AWS Toolkit for Visual Studio Codecon Step Functions.
- Consulta il riferimento AWS SAM CLI per saperne di più sulle funzionalità disponibili in. AWS SAM

Puoi anche progettare e creare flussi di lavoro in Infrastructure as Code (IaC) utilizzando strumenti di creazione visivi, come Workflow Studio in. Application Composer Per ulteriori informazioni, consulta Utilizzo di Workflow Studio in Application Composer.

Utilizzo di Workflow Studio in Application Composer

Strumento AWS per la creazione di applicazioniè un visual builder che ti aiuta a sviluppare AWS SAM AWS CloudFormation modelli utilizzando una semplice interfaccia grafica. ConApplication Composer, puoi progettare un'architettura applicativa trascinandola, raggruppandola e Servizi AWS connettendola in un'area di disegno visiva. Application Composerquindi crea un modello di infrastruttura come codice (IaC) dal tuo progetto che puoi utilizzare per distribuire l'applicazione con l'interfaccia a riga di AWS SAM comando (AWS SAMCLI) oppure. CloudFormation Per ulteriori informazioni su Application Composer, consulta Cos'è Application Composer.

Workflow Studio è disponibile Application Composer per aiutarti a progettare e creare flussi di lavoro. Workflow Studio Application Composer offre un ambiente visivo IAC che semplifica l'integrazione dei flussi di lavoro nelle applicazioni serverless create utilizzando strumenti IaC, come i modelli. CloudFormation Quando si utilizza Workflow Studio inApplication Composer, collega le singole fasi del flusso di lavoro alle AWS risorse e genera le configurazioni delle risorse in un modello. AWS SAM Aggiunge inoltre le IAM autorizzazioni necessarie per l'esecuzione del flusso di lavoro. Utilizzando Workflow Studio inApplication Composer, è possibile creare prototipi delle applicazioni e trasformarle in applicazioni pronte per la produzione.

Quando utilizzi Workflow Studio inApplication Composer, puoi passare avanti e indietro tra Application Composer canvas e Workflow Studio.

Argomenti

- Utilizzo di Workflow Studio Application Composer per creare un flusso di lavoro senza server
- Fai riferimento dinamicamente alle risorse utilizzando le sostituzioni delle CloudFormation definizioni in Workflow Studio
- Connect le attività di integrazione dei servizi a schede componenti avanzate
- Importa progetti esistenti e sincronizzali localmente
- Funzionalità di Workflow Studio non disponibili in Strumento AWS per la creazione di applicazioni

Utilizzo di Workflow Studio Application Composer per creare un flusso di lavoro senza server

1. Apri la console Application Composer e scegli Crea progetto per creare un progetto.
2. Nel campo di ricerca della palette Risorse, immetti **`state machine`**.
3. Trascina la risorsa Step FunctionsState machine sull'area di disegno.
4. Scegli Modifica in Workflow Studio per modificare la risorsa della tua macchina a stati.

   L'animazione seguente mostra come passare a Workflow Studio per modificare la definizione della macchina a stati.
   Un'animazione che illustra come utilizzare Workflow Studio inApplication Composer.

   L'integrazione con Workflow Studio per modificare le risorse delle macchine a stati create in Application Composer è disponibile solo per le `AWS::Serverless::StateMachine`risorse. Questa integrazione non è disponibile per i modelli che utilizzano la `AWS::StepFunctions::StateMachine`risorsa.

Fai riferimento dinamicamente alle risorse utilizzando le sostituzioni delle CloudFormation definizioni in Workflow Studio

In Workflow Studio, puoi utilizzare le sostituzioni di CloudFormation definizione nella definizione del flusso di lavoro per fare riferimento dinamicamente alle risorse che hai definito nel tuo modello IAc. È possibile aggiungere sostituzioni segnaposto alla definizione del flusso di lavoro utilizzando la `${dollar_sign_brace}` notazione e queste vengono sostituite con valori effettivi durante il processo di creazione dello stack. CloudFormation Per ulteriori informazioni sulle sostituzioni delle definizioni, vedere. DefinitionSubstitutions nei modelli AWS SAM

L'animazione seguente mostra come aggiungere sostituzioni segnaposto per le risorse nella definizione della macchina a stati.
Un'animazione che illustra come fare riferimento dinamicamente a risorse, ad esempio AWS Lambda funzioni, sostituzioni di definizioni quando si utilizza Workflow Studio in. Application Composer

Connect le attività di integrazione dei servizi a schede componenti avanzate

Puoi collegare le attività che richiedono integrazioni di servizi ottimizzate a schede componenti avanzate in Application Composer canvas. In questo modo vengono mappate automaticamente tutte

le sostituzioni di segnaposto specificate dalla `${dollar_sign_brace}` notazione nella definizione del flusso di lavoro e la proprietà della risorsa. `DefinitionSubstitution` `StateMachine` Aggiunge inoltre le AWS SAM politiche appropriate per la macchina a stati.

Se si mappano le attività di integrazione dei servizi ottimizzate con schede componenti standard, la linea di connessione non viene visualizzata nell'Application Composerarea di disegno.

L'animazione seguente mostra come collegare un'attività ottimizzata a una scheda componente avanzata e visualizzare le modifiche in Change Inspector.
Un'animazione che illustra come connettere attività che richiedono integrazioni di servizi ottimizzate a schede componenti avanzate quando si utilizza Workflow Studio in. Application Composer

Non puoi connettere integrazioni AWS SDK nello stato Task con schede componenti avanzate o integrazioni di servizi ottimizzate con schede componenti standard. Per queste attività, puoi mappare le sostituzioni nel pannello delle proprietà delle risorse in Application Composer canvas e aggiungere politiche nel modello. AWS SAM

> **Tip**
>
> In alternativa, puoi anche mappare le sostituzioni segnaposto per la tua macchina a stati in Definition Substitutions nel pannello delle proprietà delle risorse. Quando si esegue questa operazione, è necessario aggiungere le autorizzazioni necessarie per le chiamate di stato del Task nel ruolo di Servizio AWS esecuzione della macchina a stati. Per informazioni sulle autorizzazioni di cui potrebbe aver bisogno il tuo ruolo di esecuzione, consulta. Ruoli di esecuzione in Workflow Studio

L'animazione seguente mostra come aggiornare manualmente la mappatura sostitutiva dei segnaposto nel pannello delle proprietà della risorsa.
Un'animazione che illustra come aggiornare manualmente la mappatura sostitutiva dei segnaposto nel pannello delle proprietà delle risorse quando si utilizza Workflow Studio in. Application Composer

Importa progetti esistenti e sincronizzali localmente

Puoi aprire AWS SAM progetti esistenti CloudFormation e visualizzarli per comprenderne meglio e modificarne i design. Application Composer Utilizzando Application Composer la funzione di sincronizzazione locale, è possibile sincronizzare e salvare automaticamente i file di modello e codice sulla macchina di compilazione locale. L'utilizzo della modalità di sincronizzazione locale

può completare i flussi di sviluppo esistenti. Assicurati che il tuo browser supporti l'API File System Access, che consente alle applicazioni Web di leggere, scrivere e salvare file nel file system locale. Ti consigliamo di utilizzare Google Chrome o Microsoft Edge.

Funzionalità di Workflow Studio non disponibili in Strumento AWS per la creazione di applicazioni

Quando si utilizza Workflow Studio inApplication Composer, alcune funzionalità di Workflow Studio non sono disponibili. Inoltre, la sezione Parametri API disponibile nel ✓Inspector pannello supporta le sostituzioni delle CloudFormation definizioni. È possibile aggiungere le sostituzioni utilizzando la Modalità codice notazione. `${dollar_sign_brace}` Per ulteriori informazioni su questa notazione, vedere. DefinitionSubstitutions nei modelli AWS SAM

L'elenco seguente descrive le funzionalità di Workflow Studio che non sono disponibili quando si utilizza Workflow Studio inApplication Composer:

- Modelli Starter: i modelli Starter sono progetti di ready-to-run esempio che creano automaticamente i prototipi e le definizioni del flusso di lavoro. Questi modelli distribuiscono tutte le AWS risorse correlate di cui il progetto ha bisogno per il tuo. Account AWS
- Modalità Config: questa modalità consente di gestire la configurazione delle macchine a stati. È possibile aggiornare le configurazioni delle macchine a stati nei modelli IAc o utilizzare il pannello delle proprietà delle risorse in canvas. Application Composer Per informazioni sull'aggiornamento delle configurazioni nel pannello delle proprietà delle risorse, vedere. Connect le attività di integrazione dei servizi a schede componenti avanzate
- API TestState
- Opzione per importare o esportare le definizioni del flusso di lavoro dal pulsante a discesa Azioni in Workflow Studio. Invece, dal Application Composer menu, selezionate Apri > Cartella del progetto. Assicurati di aver abilitato la modalità di sincronizzazione locale per salvare automaticamente le modifiche nell'Application Composerarea di disegno direttamente sul computer locale.
- Pulsante Esegui. Quando utilizzi Workflow Studio inApplication Composer, Application Composer genera il codice IaC per il tuo flusso di lavoro. Pertanto, è necessario prima distribuire il modello. Quindi, esegui il flusso di lavoro nella console o tramite. AWS Command Line Interface (AWS CLI)

Creazione di una macchina a stati Lambda per Step Functions utilizzando AWS CloudFormation

Questo tutorial mostra come creare una AWS Lambda funzione di base utilizzando AWS CloudFormation. Utilizzerai la AWS CloudFormation console e un modello YAML per creare lo stack (ruoli IAM, funzione Lambda e macchina a stati). Quindi, utilizzerai la AWS Step Functions console per avviare l'esecuzione della macchina a stati.

Per ulteriori informazioni, consulta Lavorare con CloudFormation i modelli e la `AWS::StepFunctions::StateMachine` risorsa nella Guida AWS CloudFormation per l'utente.

Argomenti

- Passaggio 1: configura il AWS CloudFormation modello
- Fase 2: Usa il AWS CloudFormation modello per creare una Lambda State Machine
- Fase 3: Avviare l'esecuzione di una State Machine

Passaggio 1: configura il AWS CloudFormation modello

Prima di utilizzare i modelli di esempio, è necessario comprendere come dichiarare le diverse parti di un modello AWS CloudFormation .

Argomenti

- Per creare un ruolo IAM per Lambda
- Per creare una funzione Lambda
- Per creare un ruolo IAM per l'esecuzione della macchina a stati
- Per creare una macchina a stati Lambda

Per creare un ruolo IAM per Lambda

Definisci la policy di fiducia associata al ruolo IAM per la funzione Lambda. Gli esempi seguenti definiscono una politica di fiducia utilizzando YAML o 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"
                      }
                  ]
              }
          }
```

Per creare una funzione Lambda

Definisci le seguenti proprietà per una funzione Lambda che stamperà il messaggio. `Hello World`

> **Important**
>
> Assicurati che la tua funzione Lambda utilizzi lo stesso AWS account e la stessa AWS regione della tua macchina a stati.

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"
            }
        },
```

Per creare un ruolo IAM per l'esecuzione della macchina a stati

Definisci la politica di fiducia associata al ruolo IAM per l'esecuzione della macchina a stati.

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": "*"
                            }
                        ]
                    }
                }
            ]
        }
    },
```

Per creare una macchina a stati Lambda

Definire la macchina a stati 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"
                    ]
                }
            }
        }
```

Fase 2: Usa il AWS CloudFormation modello per creare una Lambda State Machine

Una volta compresi i componenti del AWS CloudFormation modello, puoi metterli insieme e utilizzare il modello per creare uno AWS CloudFormation stack.

Per creare la macchina a stati Lambda

1. Copiare i seguenti dati di esempio in un file denominato `MyStateMachine.yaml` per l'esempio YAML o `MyStateMachine.json` per 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": {
```

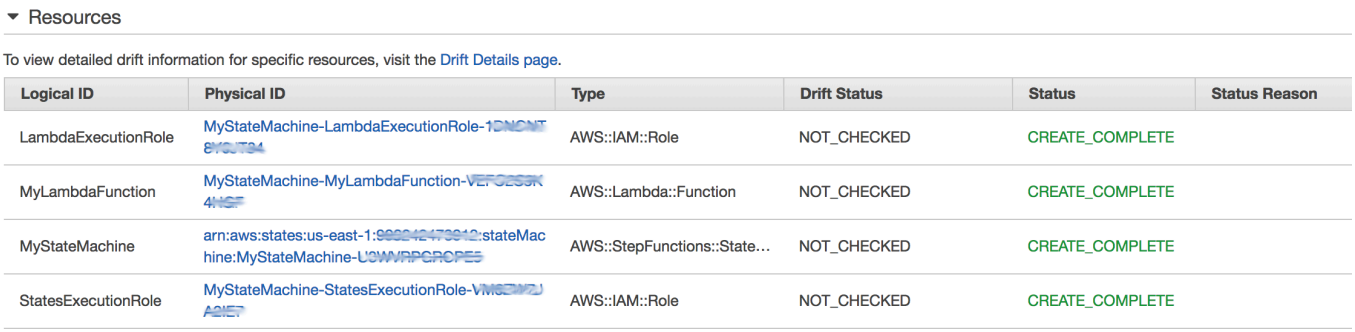
```
                "AssumeRolePolicyDocument": {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {
                                "Service": "lambda.amazonaws.com"
                            },
                            "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. Aprire la console AWS CloudFormation e scegliere Create Stack (Crea stack).
3. Nella pagina Select Template (Seleziona modello), scegli Upload a template to Amazon S3 (Carica un modello in Amazon S3). Scegliere il file `MyStateMachine` e quindi Next (Avanti).
4. Nella pagina Specificare i dettagli, per il nome dello stack, immettete**`MyStateMachine`**, quindi scegliete Avanti.
5. Nella pagina Opzioni, scegli Avanti.
6. Nella pagina Revisione, scegli Riconosco che AWS CloudFormation potrebbe creare risorse IAM. quindi scegli Crea.

   AWS CloudFormation inizia a creare lo `MyStateMachine` stack e visualizza lo stato CREATE_IN_PROGRESS. Al termine del processo, AWS CloudFormation mostra lo stato CREATE_COMPLETE.
7. (Facoltativo) Per visualizzare le risorse nello stack, selezionare lo stack e scegliere la scheda Resources (Risorse).



Fase 3: Avviare l'esecuzione di una State Machine

Dopo aver creato la macchina a stati Lambda, puoi avviarne l'esecuzione.

Per avviare l'esecuzione della macchina a stati

1. Apri la console Step Functions e scegli il nome della macchina a stati che hai creato utilizzando AWS CloudFormation.
2. Nella pagina ***MyStateMachine-ABCDEFGHIJ1K***, scegli Nuova esecuzione.

   Viene visualizzata la pagina New execution (Nuova esecuzione).
3. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

   > **Note**
   >
   > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.
4. Selezionare Start Execution (Avvia esecuzione).

   Viene avviata una nuova esecuzione della macchina a stati e visualizzata una pagina inedita con l'esecuzione in corso.
5. (Facoltativo) In Execution Details (Dettagli esecuzione), esaminare lo Execution Status (Stato esecuzione) e i timestamp Started (Avviato) e Closed (Chiuso).
6. Per visualizzare i risultati dell'esecuzione, selezionare Output (Output).

Creazione di una macchina a Lambda stati per Step Functions l'utilizzo AWS CDK

Questo tutorial mostra come creare una macchina a AWS Step Functions stati che contenga una AWS Lambda funzione utilizzando. AWS Cloud Development Kit (AWS CDK) AWS CDKSi tratta di un framework Infrastructure as Code (IAC) che consente di definire l' AWS infrastruttura utilizzando

un linguaggio di programmazione completo. Puoi scrivere un'app in una delle lingue supportate dall'app contenente uno o più stack. CDK Quindi, puoi sintetizzarla in un AWS CloudFormation modello e distribuirla nel tuo account. AWS Useremo questo metodo per definire una macchina a Step Functions stati contenente una Lambda funzione, quindi useremo il AWS Management Console per eseguire la macchina a stati.

Prima di iniziare questo tutorial, è necessario configurare l'ambiente di AWS CDK sviluppo come descritto in Guida introduttiva ai prerequisiti della AWS CDK Guida per gli AWS Cloud Development Kit (AWS CDK) sviluppatori. Quindi, installalo AWS CDK con il seguente comando su: AWS CLI

```
npm install -g aws-cdk
```

Questo tutorial produce lo stesso risultato dithe section called “Creazione di una macchina a stati Lambda utilizzando AWS CloudFormation”. Tuttavia, in questo tutorial, AWS CDK non è necessario che tu IAM crei alcun ruolo; lo AWS CDK fa per te. La AWS CDK versione include anche un Succeed passaggio per illustrare come aggiungere ulteriori passaggi alla macchina a stati.

> **Tip**
>
> Per distribuire un'applicazione serverless di esempio che avvia un Step Functions flusso di lavoro utilizzando AWS CDK with TypeScript to your Account AWS, consulta il Modulo 10 - Distribuisci con AWS CDK di The Workshop. AWS Step Functions

Argomenti

- Fase 1: Configurazione del progetto AWS CDK
- Passaggio 2: Utilizzare AWS CDK per creare una macchina a stati
- Fase 3: Avviare l'esecuzione di una macchina a stati
- Fase 4: eliminazione
- Passaggi successivi

Fase 1: Configurazione del progetto AWS CDK

1. Nella tua home directory, o in un'altra directory se preferisci, esegui il comando seguente per creare una directory per la tua nuova app. AWS CDK

> **Important**
>
> Assicurati di assegnare un nome alla directory`step`. Il modello di AWS CDK applicazione utilizza il nome della directory per generare nomi per i file e le classi di origine. Se utilizzi un nome diverso, l'app non corrisponderà a questo tutorial.

TypeScript

```
mkdir step && cd step
```

JavaScript

```
mkdir step && cd step
```

Python

```
mkdir step && cd step
```

Java

```
mkdir step && cd step
```

C#

Assicurati di aver installato.NET versione 6.0 o successiva. Per informazioni, consulta Versioni supportate.

```
mkdir step && cd step
```

2. Inizializza l'app utilizzando il comando cdk init. Specificate il modello («app») e il linguaggio di programmazione desiderati come mostrato negli esempi seguenti.

TypeScript

```
cdk init --language typescript
```

JavaScript

```
cdk init --language javascript
```

Python

```
cdk init --language python
```

Dopo l'inizializzazione del progetto, attiva l'ambiente virtuale del progetto e installa le dipendenze AWS CDK di base.

```
source .venv/bin/activate
python -m pip install -r requirements.txt
```

Java

```
cdk init --language java
```

C#

```
cdk init --language csharp
```

Passaggio 2: Utilizzare AWS CDK per creare una macchina a stati

Per prima cosa, presenteremo le singole parti di codice che definiscono la Lambda funzione e la macchina a Step Functions stati. Quindi, spiegheremo come metterli insieme nella tua AWS CDK app. Infine, vedrete come sintetizzare e distribuire queste risorse.

Per creare una funzione Lambda

Il AWS CDK codice seguente definisce la Lambda funzione, fornendo il codice sorgente in linea.

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)
});
```

Puoi vedere in questo breve codice di esempio:

- Il nome logico della funzione,`MyLambdaFunction`.
- Il codice sorgente della funzione, incorporato come stringa nel codice sorgente dell'AWS CDKapp.
- Altri attributi della funzione, come il runtime da utilizzare (Nodo 18.x), il punto di ingresso della funzione e un timeout.

Per creare una macchina a stati

La nostra macchina a stati ha due stati: una Lambda funzione, un'attività e uno Succeed stato. La funzione richiede che creiamo una funzione Step Functions the section called “Attività” che richiami la nostra funzione. Questo stato dell'attività viene utilizzato come primo passaggio nella macchina a stati. Lo stato di successo viene aggiunto alla macchina a stati utilizzando il `next()` metodo dello stato Task. Il codice seguente richiama innanzitutto la funzione denominata`MyLambdaTask`, quindi utilizza il `next()` metodo per definire uno stato di successo denominato. `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")))
});
```

Per creare e distribuire l'app AWS CDK

Nel AWS CDK progetto appena creato, modifica il file che contiene la definizione dello stack in modo che assomigli al seguente codice di esempio. Riconoscerai le definizioni della Lambda funzione e della macchina a Step Functions stati delle sezioni precedenti.

1. Aggiorna lo stack come mostrato negli esempi seguenti.

   TypeScript

   Aggiorna `lib/step-stack.ts` con il codice seguente.

```
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

   Aggiorna `lib/step-stack.js` con il seguente codice.

```
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

Aggiorna `step/step_stack.py` con il seguente codice.

```
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

Aggiorna `src/main/java/com.myorg/StepStack.java` con il seguente codice.

```
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#

Aggiorna `scr/Step/StepStack.cs` con il seguente codice.

```
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. Salva il file sorgente, quindi esegui il `cdk synth` comando nella directory principale dell'app.

   AWS CDKesegue l'app e sintetizza un AWS CloudFormation modello a partire da essa. AWS CDKquindi visualizza il modello.

   > **Note**
   >
   > Se lo hai usato TypeScript per creare il tuo AWS CDK progetto, l'esecuzione del `cdk synth` comando potrebbe restituire il seguente errore.
   >
   > ```
   > TSError: # Unable to compile TypeScript:
   > bin/step.ts:7:33 - error TS2554: Expected 2 arguments, but got 3.
   > ```
   >
   > Modificate il `bin/step.ts` file come mostrato nell'esempio seguente per risolvere questo errore.
   >
   > ```
   > #!/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. Per distribuire la funzione Lambda e la macchina a stati Step Functions AWS sul tuo account, emetti. `cdk deploy` Ti verrà chiesto di approvare le policy IAM che ha generato. AWS CDK

Fase 3: Avviare l'esecuzione di una macchina a stati

Dopo aver creato la macchina a stati, è possibile avviarne l'esecuzione.

Per avviare l'esecuzione della macchina a stati

1. Apri la console Step Functions e scegli il nome della macchina a stati che hai creato utilizzandoAWS CDK.
2. Nella pagina della macchina a stati, scegli Avvia esecuzione.

   Viene visualizzata la finestra di dialogo Avvia esecuzione.
3. (Facoltativo) Per identificare l'esecuzione, è possibile specificare un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

   > **Note**
   >
   > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.
4. Selezionare Start Execution (Avvia esecuzione).

   L'esecuzione della macchina a stati inizia e viene visualizzata una nuova pagina che mostra l'esecuzione in esecuzione.
5. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Fase 4: eliminazione

Dopo aver testato la tua macchina a stati, ti consigliamo di rimuovere sia la macchina a stati che la relativa funzione Lambda per liberare risorse nel tuo. Account AWS Esegui il `cdk destroy` comando nella directory principale dell'app per rimuovere la macchina a stati.

Passaggi successivi

Per ulteriori informazioni sullo sviluppo AWS dell'infrastruttura utilizzandoAWS CDK, consulta la Guida per gli AWS CDK sviluppatori.

Per informazioni sulla scrittura di app AWS CDK nel tuo linguaggio preferito, consulta:

TypeScript

- Lavorare con AWS CDK in TypeScript

JavaScript

- Lavorare con AWS CDK in JavaScript

Python

- Lavorare con AWS CDK Python

Java

- Lavorare con AWS CDK Java

C#

- Lavorare con AWS CDK C#

Per ulteriori informazioni sui moduli AWS Construct Library utilizzati in questo tutorial, consultate le seguenti panoramiche di riferimento sulle AWS CDK API:

- aws-lambda
- funzioni aws-step
- aws-stepfunctions-tasks

Creazione di un'API Gateway REST con Synchronous Express State Machine utilizzando AWS CDK

Questo tutorial mostra come creare un'API API Gateway REST con Synchronous Express State Machine come integrazione di backend utilizzando. AWS Cloud Development Kit (AWS CDK) Questo tutorial utilizzerà il `StepFunctionsRestApi` costrutto per connettere la State Machine all'API Gateway. Il `StepFunctionsRestApi` costrutto imposterà una mappatura di input/output predefinita e l'API REST di API Gateway, con le autorizzazioni richieste e un metodo HTTP «ANY». AWS CDK Si tratta di un framework Infrastructure as Code (IAC) che consente di definire AWS l'infrastruttura utilizzando un linguaggio di programmazione completo. Scrivi un'app in uno dei linguaggi supportati dal CDK, contenente uno o più stack, quindi la sintetizzi in un AWS CloudFormation modello e la distribuisci sul tuo account. AWS Lo useremo per definire un'API API Gateway REST, che è integrata con Synchronous Express State Machine come backend, quindi utilizzeremo AWS Management Console per avviare l'esecuzione.

Prima di iniziare questo tutorial, configura il tuo ambiente di AWS CDK sviluppo come descritto in Getting Started With the AWS CDK - Prerequisiti, quindi installa il seguente comando: AWS CDK

```
npm install -g aws-cdk
```

Argomenti

- Fase 1: Configura il tuo progetto AWS CDK
- Passaggio 2: utilizzare AWS CDK per creare un'API API Gateway REST con integrazione backend Synchronous Express State Machine
- Fase 3: Testare l'API Gateway
- Fase 4: eliminazione

Fase 1: Configura il tuo progetto AWS CDK

Innanzitutto, crea una directory per la tua nuova AWS CDK app e inizializza il progetto.

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
```

Dopo l'inizializzazione del progetto, attiva l'ambiente virtuale del progetto e installa le dipendenze AWS CDK di base.

```
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**
>
> Assicurati di assegnare un nome alla directory. `stepfunctions-rest-api` Il modello di AWS CDK applicazione utilizza il nome della directory per generare nomi per i file e le classi di origine. Se utilizzi un nome diverso, l'app non corrisponderà a questo tutorial.

Ora installa i moduli della libreria di costruzione per AWS Step Functions 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

Modifica il file `pom.xml` del progetto per aggiungere le seguenti dipendenze all'interno del container `<dependencies>`.

```
        <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 installa automaticamente queste dipendenze la prossima volta che crei la tua app. Per eseguire la creazione, emetti `mvn compile` o utilizza il comando Build dell'IDE Java.

C#

```
dotnet add src/StepfunctionsRestApi package Amazon.CDK.AWS.Stepfunctions
dotnet add src/StepfunctionsRestApi package Amazon.CDK.AWS.APIGateway
```

Puoi anche installare i pacchetti indicati utilizzando la NuGet GUI di Visual Studio, disponibile tramite Tools > NuGet Package Manager > Manage NuGet Packages for Solution.

Dopo aver installato i moduli, puoi usarli nell' AWS CDK app importando i seguenti pacchetti.

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

Aggiungi quanto segue all'`import`interno`stepfunctions-rest-api.go`.

```
"github.com/aws/aws-cdk-go/awscdk/awsapigateway"
"github.com/aws/aws-cdk-go/awscdk/awsstepfunctions"
```

Passaggio 2: utilizzare AWS CDK per creare un'API API Gateway REST con integrazione backend Synchronous Express State Machine

Per prima cosa, presenteremo le singole parti di codice che definiscono la Synchronous Express State Machine e l'API API Gateway REST, quindi spiegheremo come metterle insieme nella tua AWS CDK app. Poi vedrete come sintetizzare e distribuire queste risorse.

> **Note**
>
> La State Machine che mostreremo qui sarà una semplice State Machine con uno stato. `Pass`

Per creare una Express State Machine

Questo è il AWS CDK codice che definisce una semplice macchina a stati con uno `Pass` stato.

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,
})
```

In questo breve frammento puoi vedere:

- La definizione della macchina denominata`PassState`, che è uno `Pass` Stato.
- Il nome logico della State Machine,`MyStateMachine`.
- La definizione della macchina viene utilizzata come definizione della macchina a stati.
- Il tipo di macchina a stati è impostato in `StepFunctionsRestApi` modo `EXPRESS` da consentire solo una macchina a stati Synchronous Express.

Per creare l'API REST di API Gateway utilizzando **`StepFunctionsRestApi`** construct

Useremo `StepFunctionsRestApi` construct per creare l'API REST API Gateway con le autorizzazioni richieste e la mappatura di input/output predefinita.

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,
})
```

Per creare e distribuire l'app AWS CDK

Nel AWS CDK progetto che hai creato, modifica il file contenente la definizione dello stack in modo che assomigli al codice seguente. Riconoscerai le definizioni della macchina a stati Step Functions e dell'API Gateway dall'alto.

TypeScript

Aggiorna `lib/stepfunctions-rest-api-stack.ts` per leggere quanto riportato di seguito.

```
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

Aggiorna `lib/stepfunctions-rest-api-stack.js` per leggere quanto riportato di seguito.

```
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

Aggiorna `stepfunctions_rest_api/stepfunctions_rest_api_stack.py` per leggere quanto riportato di seguito.

```
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

Aggiorna `src/main/java/com.myorg/StepfunctionsRestApiStack.java` per leggere quanto riportato di seguito.

```
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#

Aggiorna `src/StepfunctionsRestApi/StepfunctionsRestApiStack.cs` per leggere quanto riportato di seguito.

```
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

Aggiorna `stepfunctions-rest-api.go` per leggere quanto riportato di seguito.

```
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
```

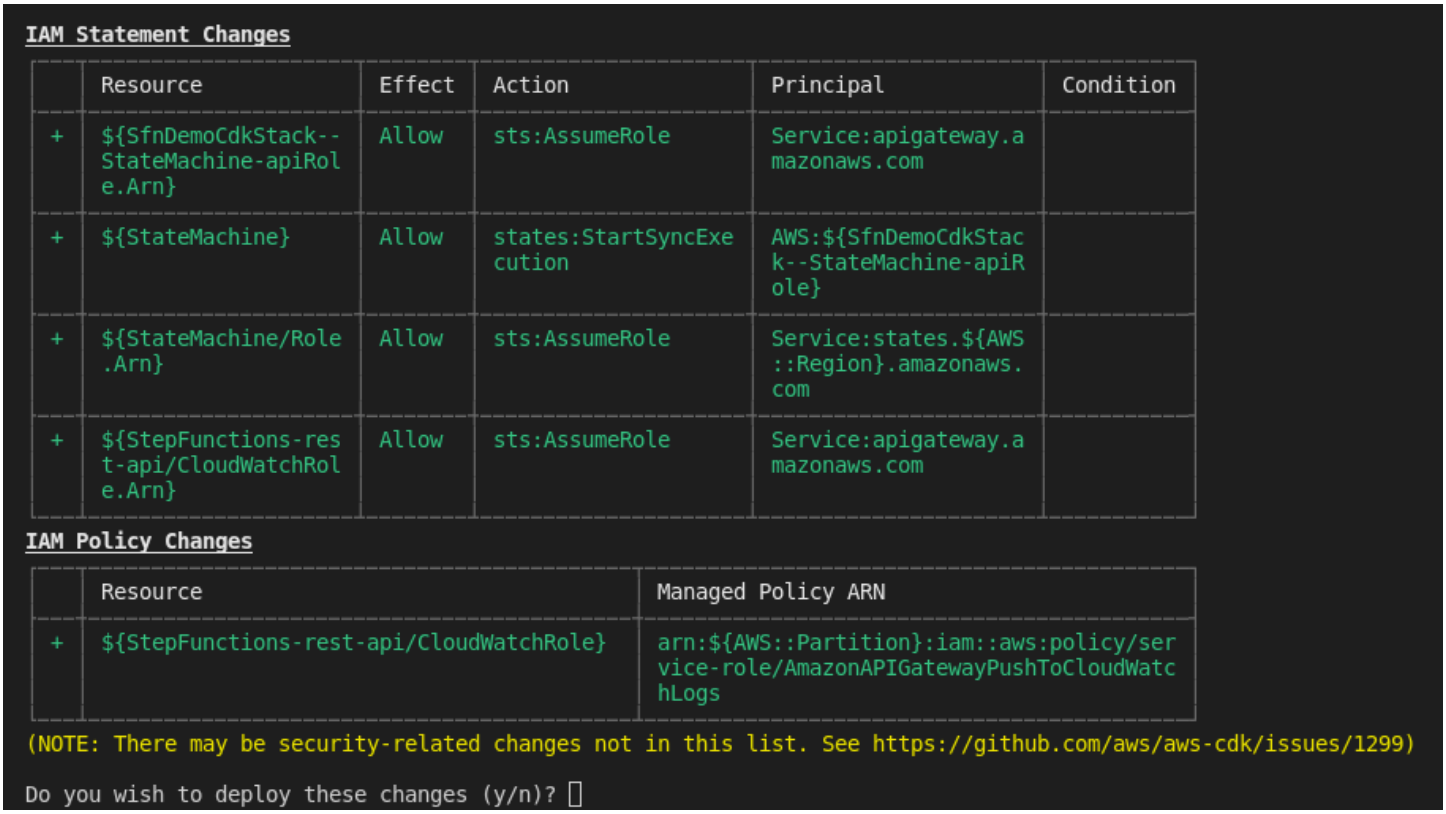
```
// be deployed. For more information see: https://docs.aws.amazon.com/cdk/latest/
guide/environments.html
func env() *awscdk.Environment {
    // If unspecified, this stack will be "environment-agnostic".
    // Account/Region-dependent features and context lookups will not work, but a
    // single 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")),
    // }
}
```

Salva il file di origine, quindi emetti `cdk synth` nella directory principale dell'app. AWS CDK Esegue l'app e sintetizza un AWS CloudFormation modello a partire da essa, quindi visualizza il modello.

Per distribuire effettivamente Amazon API Gateway e la macchina a AWS Step Functions stati sul tuo account AWS, `cdk deploy` emetti. Ti verrà chiesto di approvare le policy IAM che AWS CDK ha generato. Le politiche che verranno create avranno un aspetto simile a questo:



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)

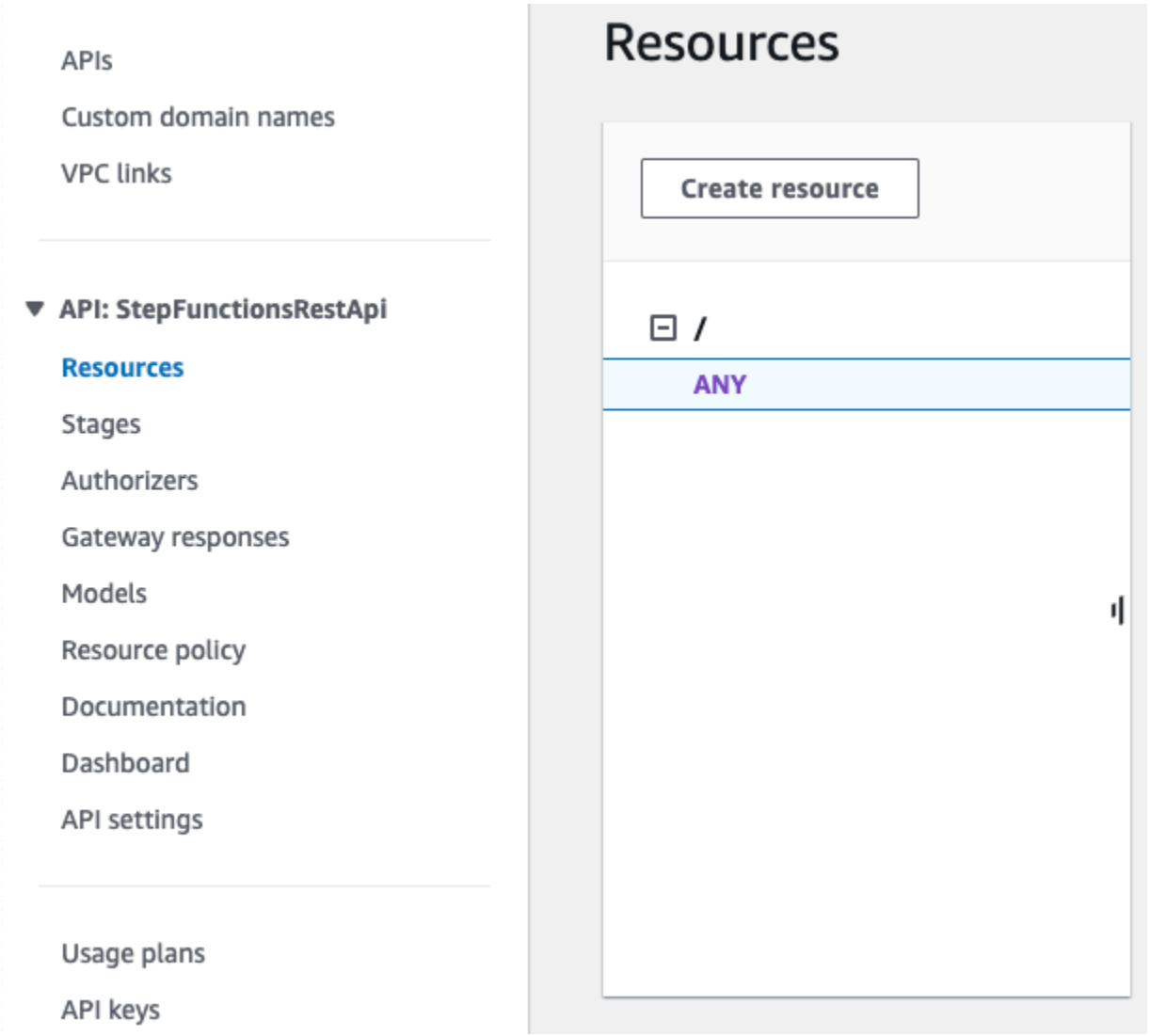
Do you wish to deploy these changes (y/n)?

Fase 3: Testare l'API Gateway

Dopo aver creato l'API REST di API Gateway con Synchronous Express State Machine come integrazione di backend, puoi testare l'API Gateway.

Per testare l'API Gateway distribuito utilizzando la console API Gateway

1. Apri la console Amazon API Gateway e accedi.
2. Scegli il nome della tua API REST`StepFunctionsRestApi`.
3. Nel riquadro Risorse, scegli il `ANY` metodo.



4. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
5. In Method (Metodo) seleziona POST.
6. Per il corpo della richiesta, copia i seguenti parametri della richiesta.

```
{
    "key": "Hello"
}
```

7. Scegli Test (Esegui test). Verranno visualizzate le seguenti informazioni:
   - Request (Richiesta) è il percorso della risorsa chiamato per il metodo.
   - Status (Stato) è il codice dello stato HTTP della risposta.

- Latency (Latenza) è l'intervallo di tempo tra la ricezione della richiesta dall'intermediario e la risposta restituita.
- Corpo della risposta è il corpo della risposta HTTP.
- Intestazioni delle risposte sono le intestazioni di risposta HTTP.
- Il log mostra le voci simulate di Amazon CloudWatch Logs che sarebbero state scritte se questo metodo fosse stato chiamato al di fuori della console API Gateway.

> **Note**
>
> Sebbene le voci di CloudWatch Logs siano simulate, i risultati della chiamata al metodo sono reali.

L'output del corpo di Response dovrebbe essere qualcosa del genere:

```
"Hello"
```

> **Tip**
>
> Prova l'API Gateway con metodi diversi e un input non valido per visualizzare l'output dell'errore. Potresti voler cambiare la macchina a stati per cercare una chiave particolare e durante il test fornire la chiave sbagliata per fallire l'esecuzione della State Machine e generare un messaggio di errore nell'output del corpo della risposta.

Per testare l'API distribuita tramite cURL

1. Aprire una finestra del terminale.
2. Copiare il comando cURL seguente e incollarlo nella finestra del terminale, sostituendo `<api-id>` con l'ID dell'API e `<region>` con la regione in cui l'API è distribuita.

```
curl -X POST\
 'https://<api-id>.execute-api.<region>.amazonaws.com/prod' \
 -d '{"key":"Hello"}' \
 -H 'Content-Type: application/json'
```

L'output del Response Body dovrebbe essere qualcosa del genere:

```
"Hello"
```

> **Tip**
>
> Prova l'API Gateway con metodi diversi e un input non valido per visualizzare l'output dell'errore. Potresti voler cambiare la macchina a stati per cercare una chiave particolare e durante il test fornire la chiave sbagliata per fallire l'esecuzione della State Machine e generare un messaggio di errore nell'output del Response Body.

Fase 4: eliminazione

Quando hai finito di provare il tuo API Gateway, puoi smontare sia la macchina a stati che l'API Gateway utilizzando il CDK AWS. Emetti `cdk destroy` nella directory principale della tua app.

AWS Step Functions SDK per Data Science per Python

AWS Step Functions Data Science SDK è una libreria open source per data scientist. Con questo SDK, puoi creare flussi di lavoro che elaborano e pubblicano modelli di machine learning utilizzando SageMaker e Step Functions. Puoi anche creare flussi di lavoro di machine learning in più fasi in Python che orchestrano l' AWS infrastruttura su larga scala, senza dover fornire e integrare i servizi separatamente. AWS

Il AWS Step Functions Data Science SDK fornisce un'API Python in grado di creare e richiamare flussi di lavoro Step Functions. Puoi gestire ed eseguire questi flussi di lavoro direttamente in Python e nei notebook Jupyter.

Oltre a creare flussi di lavoro pronti per la produzione direttamente in Python, AWS Step Functions Data Science SDK consente di copiare quel flusso di lavoro, sperimentare nuove opzioni e quindi mettere in produzione il flusso di lavoro perfezionato.

Per ulteriori informazioni su AWS Step Functions Data Science SDK, consulta quanto segue:

- Progetto ONNX su GitHub
- Documentazione sugli SDK

- I seguenti notebook di esempio, disponibili nelle istanze di notebook Jupyter nella console e nel relativo progetto: SageMaker GitHub
  - `hello_world_workflow.ipynb`
  - `machine_learning_workflow_abalone.ipynb`
  - `training_pipeline_pytorch_mnist.ipynb`

Distribuzione di macchine a stati utilizzando Terraform

Terraform by HashiCorp è un framework per la creazione di applicazioni che utilizzano l'infrastruttura come codice (IaC). Con Terraform, puoi creare macchine a stati e utilizzare funzionalità, come l'anteprima delle implementazioni dell'infrastruttura e la creazione di modelli riutilizzabili. I modelli Terraform ti aiutano a mantenere e riutilizzare il codice suddividendolo in blocchi più piccoli.

Se conosci Terraform, puoi seguire il ciclo di vita di sviluppo descritto in questo argomento come modello per la creazione e la distribuzione delle tue macchine a stati in Terraform. Se non hai familiarità con Terraform, ti consigliamo di completare prima il workshop Introduzione a Terraform per familiarizzare con Terraform. AWS

> **Tip**
>
> Per distribuire un esempio di macchina a stati creata utilizzando Terraform sul tuo computerAccount AWS, consulta il modulo Gestire macchine a stati con infrastruttura come codice di The Workshop. AWS Step Functions

In questo argomento

- Prerequisiti
- Ciclo di vita di sviluppo di macchine a stati con Terraform
- Ruoli e politiche IAM per la tua macchina a stati

Prerequisiti

Prima di iniziare, assicurati di completare i seguenti prerequisiti:

- Installa Terraform sul tuo computer. Per informazioni sull'installazione di Terraform, consulta Installa Terraform.

- Installa Step Functions Local sul tuo computer. Ti consigliamo di installare l'immagine Docker di Step Functions Local per utilizzare Step Functions Local. Per ulteriori informazioni, consulta Test di macchine a stato locale.
- Installa AWS SAM CLI. Per informazioni sull'installazione, consulta Installazione della AWS SAM CLI nella Guida per gli AWS Serverless Application Modelsviluppatori.
- Installa AWS Toolkit for Visual Studio Code per visualizzare il diagramma del flusso di lavoro delle tue macchine a stati. Per informazioni sull'installazione, vedere Installazione di AWS Toolkit for Visual Studio Code nella Guida per l'AWS Toolkit for Visual Studio Codeutente.

Ciclo di vita di sviluppo di macchine a stati con Terraform

La procedura seguente spiega come utilizzare un prototipo di macchina a stati creato utilizzando Workflow Studio nella console Step Functions come punto di partenza per lo sviluppo locale con Terraform e il. AWS Toolkit for Visual Studio Code

Per visualizzare l'esempio completo che illustra lo sviluppo di macchine a stati con Terraform e presenta le migliori pratiche in dettaglio, vedere Best practices for writing Step Functions Terraform projects.

Per avviare il ciclo di vita di sviluppo di una macchina a stati con Terraform

1. Avvia un nuovo progetto Terraform con il seguente comando.

   ```
   terraform init
   ```

2. Apri la console Step Functions per creare un prototipo per la tua macchina a stati.
3. In Workflow Studio, effettuate le seguenti operazioni:

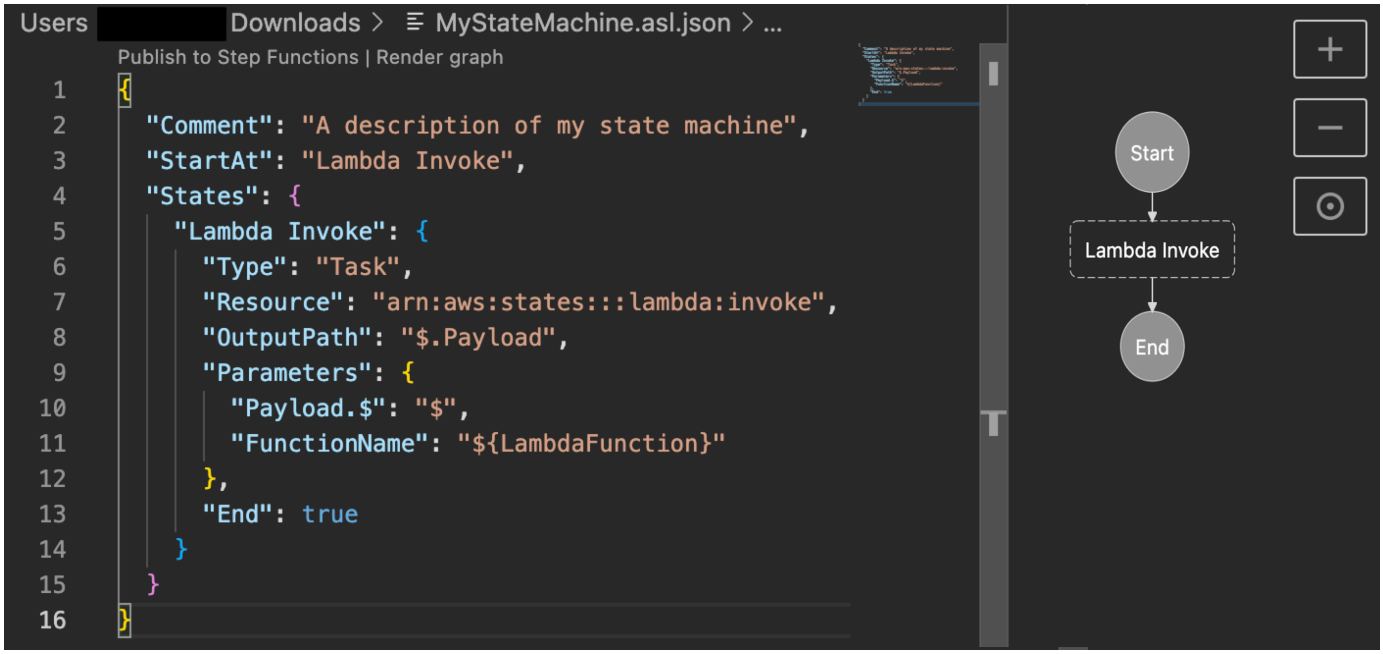
   a. Crea il prototipo del tuo flusso di lavoro.

   b. Esporta la definizione Amazon States Language (ASL) del tuo flusso di lavoro. Per fare ciò, scegli l'elenco a discesa Importa/Esporta, quindi seleziona Esporta definizione JSON.
4. Salva la definizione ASL esportata nella directory del progetto.

   Si passa la definizione ASL esportata come parametro di input alla risorsa `aws_sfn_state_machine`Terraform che utilizza la funzione. `templatefile` Questa funzione viene utilizzata all'interno del campo di definizione che passa la definizione ASL esportata e qualsiasi sostituzione di variabile.

> **Tip**
>
> Poiché il file di definizione ASL può contenere lunghi blocchi di testo, si consiglia di evitare il metodo EOF in linea. Ciò semplifica la sostituzione dei parametri nella definizione della macchina a stati.

5. (Facoltativo) Aggiorna la definizione ASL all'interno del tuo IDE e visualizza le modifiche utilizzando. AWS Toolkit for Visual Studio Code



Per evitare di esportare continuamente la definizione e rifattorizzarla nel progetto, ti consigliamo di apportare aggiornamenti localmente nel tuo IDE e di tenere traccia di questi aggiornamenti con Git.

6. Testa il tuo flusso di lavoro utilizzando Step Functions Local.

> **Tip**
>
> Puoi anche testare localmente le integrazioni dei servizi con le funzioni Lambda e le API API Gateway nella tua macchina a stati utilizzando AWS SAM CLI Local.

7. Visualizza l'anteprima della tua macchina a stati e di altre AWS risorse prima di distribuirla. Per farlo, esegui il comando seguente.

```
terraform plan
```

8. Distribuisci la tua macchina a stati dall'ambiente locale o tramite pipeline CI/CD utilizzando il seguente comando.

```
terraform apply
```

9. (Facoltativo) Pulisci le tue risorse ed elimina la macchina a stati utilizzando il seguente comando.

```
terraform destroy
```

Ruoli e politiche IAM per la tua macchina a stati

Utilizza le politiche di integrazione del servizio Terraform per aggiungere le autorizzazioni IAM necessarie alla tua macchina a stati, ad esempio l'autorizzazione a richiamare le funzioni Lambda. Puoi anche definire ruoli e politiche espliciti e associarli alla tua macchina a stati.

Il seguente esempio di policy IAM concede alla tua macchina a stati l'accesso per richiamare una funzione Lambda denominata. *myFunction*

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunction"
      ],
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:myFunction"
    }
  ]
}
```

Ti consigliamo inoltre di utilizzare l'origine `aws_iam_policy_document`dati per definire le politiche IAM per le tue macchine a stati in Terraform. Questo ti aiuta a verificare se la tua politica non è valida e a sostituire le risorse con variabili.

Il seguente esempio di policy IAM utilizza l'origine `aws_iam_policy_document` dati e concede alla macchina a stati l'accesso per richiamare una funzione Lambda denominata. *myFunction*

```
data "aws_iam_policy_document" "state_machine_role_policy" {

  statement {
    effect = "Allow"

    actions = [
      "lambda:InvokeFunction"
    ]

    resources = ["${aws_lambda_function.[[myFunction]].arn}:*"]
  }

}
```

> **Tip**
>
> Per visualizzare modelli AWS architettonici più avanzati implementati con Terraform, consulta gli esempi di Terraform su Serverless Land Workflows Collection.

Esecuzione di test e debugging

Step Functionsoffre diversi modi per testare ed eseguire il debug delle macchine a stati. Ad esempio, è possibile testare ed eseguire il debug delle macchine a stati nella console, utilizzare l'TestStateAPI per testare un singolo stato o utilizzare Step Functions Local per testare le macchine a stati localmente.

Utilizzando l'TestStateAPI, si fornisce la definizione di un singolo stato e lo si esegue. È possibile testare un singolo stato senza creare una macchina a stati o aggiornare una macchina a stati esistente.

Step FunctionsLocal è una versione scaricabile Step Functions che consente di sviluppare e testare applicazioni utilizzando una versione in Step Functions esecuzione nel proprio ambiente di sviluppo. Con Step Functions Local, puoi eseguire le tue macchine statali per testarne i flussi di dati di input e output, le integrazioni con i servizi supportati e altro ancora nel tuo ambiente di sviluppo locale.

Argomenti

- Utilizzo TestState dell'API per testare uno stato
- Test di macchine a stato locale

Utilizzo TestState dell'API per testare uno stato

L'TestStateAPI accetta la definizione di un singolo stato e la esegue. È possibile testare uno stato senza creare una macchina a stati o aggiornare una macchina a stati esistente.

Utilizzando l' TestState API, puoi testare quanto segue:

- Flusso di dati di elaborazione in ingresso e in uscita di uno stato.
- Un'Servizio AWS integrazione con altre Servizi AWS richieste e risposte
- Una richiesta e una risposta HTTP Task

Per testare uno stato, puoi anche utilizzare la Step Functionsconsole o AWS Command Line Interface (AWS CLI)l'SDK.

L'`TestState`API presuppone un ruolo IAM che deve contenere le IAM autorizzazioni necessarie per le risorse a cui accede lo stato. Per informazioni sulle autorizzazioni di cui uno stato potrebbe aver bisogno, consulta. IAMautorizzazioni per l'utilizzo dell'API TestState

Argomenti

- Considerazioni sull'utilizzo dell'API TestState
- Utilizzo dei livelli di ispezione nell'API TestState
- IAMautorizzazioni per l'utilizzo dell'API TestState
- Test di uno stato (console)
- Testare uno stato utilizzando AWS CLI
- Test e debug del flusso di dati di input e output

Considerazioni sull'utilizzo dell'API TestState

Utilizzando l'TestStateAPI, puoi testare solo uno stato alla volta. Gli stati che puoi testare includono quanto segue:

- Tutti i tipi di attività, ad eccezione delle attività
- Pass
- Attendi
- Choice
- Succeed
- Fail

Durante l'utilizzo dell'`TestState`API, tieni presente le seguenti considerazioni.

- L' TestState API non include il supporto per quanto segue:
  - Attivitàafferma che utilizzano i seguenti tipi di risorse:
    - Attività
    - modelli di integrazione dei servizi di tipo `.sync` o `.waitForTaskToken`
  - Parallelstato
  - Eseguire la mappaturastato
- Un test può durare fino a cinque minuti. Se un test supera questa durata, fallisce con l'`States.Timeout`errore.

Utilizzo dei livelli di ispezione nell'API TestState

Per testare uno stato utilizzando l'TestStateAPI, devi fornire la definizione di quello stato. Il test restituisce quindi un output. Per ogni stato, è possibile specificare la quantità di dettagli che si desidera visualizzare nei risultati del test. Questi dettagli forniscono informazioni aggiuntive sullo stato che stai testando. Ad esempio, se hai utilizzato filtri di elaborazione dei dati di input e output, come `InputPath`o `ResultPath`in uno stato, puoi visualizzare i risultati intermedi e finali dell'elaborazione dei dati.

Step Functionsfornisce i seguenti livelli per specificare i dettagli da visualizzare:

- INFORMAZIONI
- ESEGUIRE IL DEBUG
- TRACCIA

Tutti questi livelli restituiscono anche `nextState` i campi `status` and. `status`indica lo stato dell'esecuzione dello stato. Ad esempio`SUCCEEDED`, `FAILED`, `RETRIABLE`, e`CAUGHT_ERROR`. `nextState`indica il nome dello stato successivo verso cui passare. Se non hai definito uno stato successivo nella definizione, questo campo restituisce un valore vuoto.

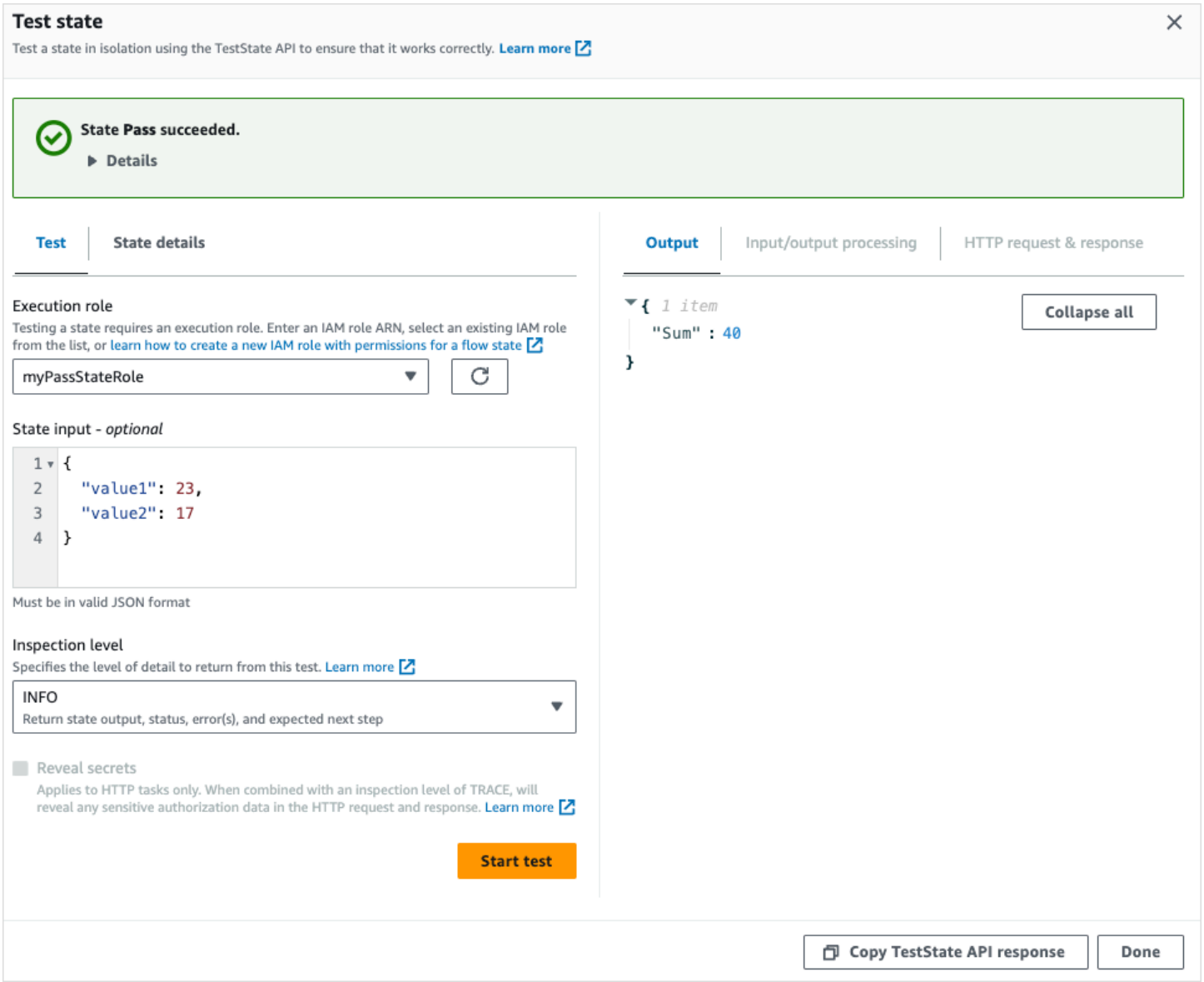
Per informazioni sul test di uno stato utilizzando questi livelli di ispezione nella Step Functions console eAWS CLI, consulta Test di uno stato (console) eTestare uno stato utilizzando AWS CLI.

INFO Inspection Level

Se il test ha esito positivo, questo livello mostra l'output dello stato. Se il test fallisce, questo livello mostra l'output dell'errore. Per impostazione predefinita, Step Functions imposta il livello di ispezione su INFO se non si specifica un livello.

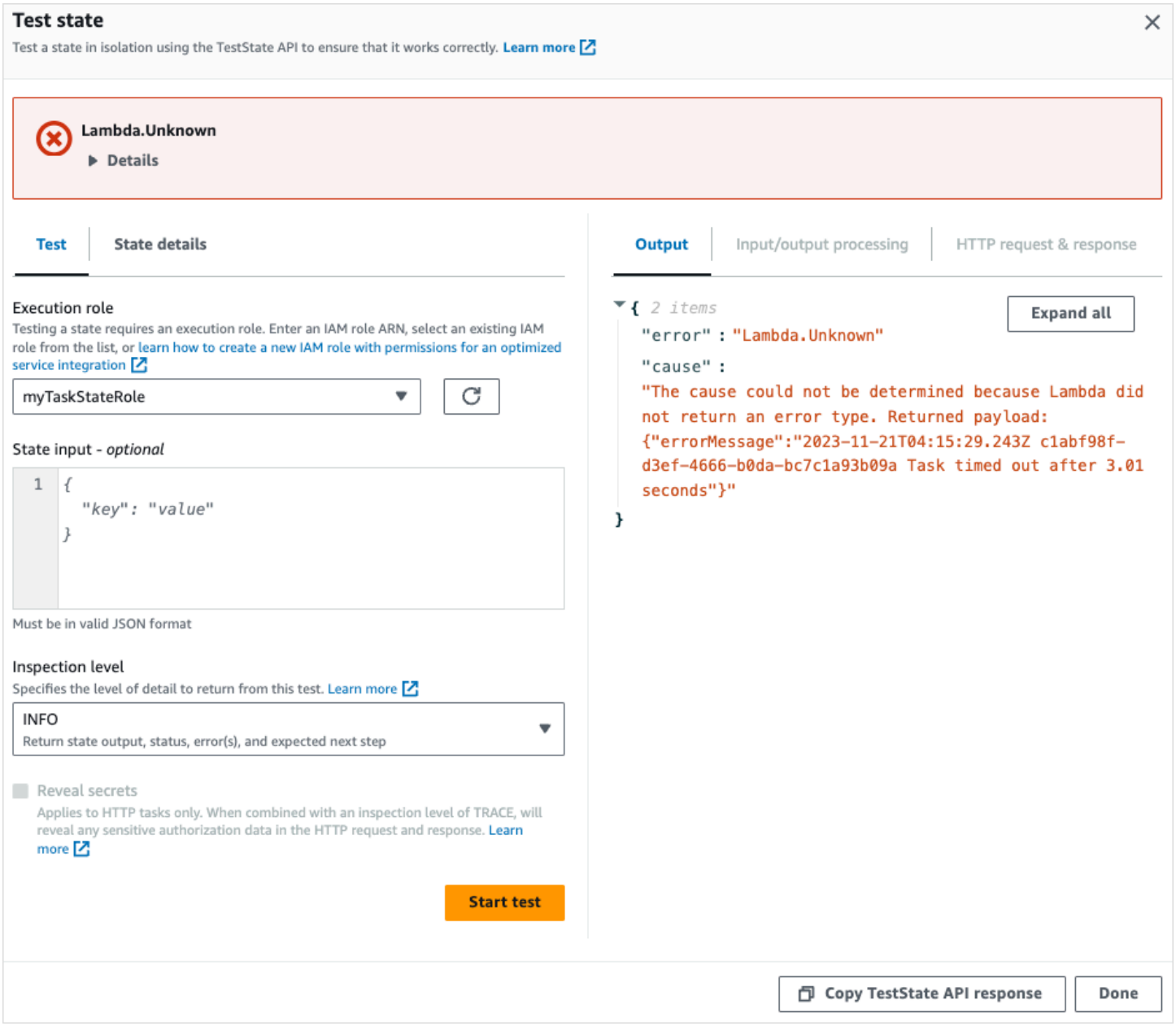
Esempio di test con livello INFO che ha esito positivo

L'immagine seguente mostra un test per uno stato Pass che ha esito positivo. Il livello di ispezione per questo stato è impostato su INFO e l'output per lo stato viene visualizzato nella scheda Output.



Esempio di test con livello INFO che fallisce

L'immagine seguente mostra un test fallito per uno stato Task quando il livello di ispezione è impostato su INFO. La scheda Output mostra l'output dell'errore che include il nome dell'errore e una spiegazione dettagliata della causa dell'errore.



Livello di ispezione DEBUG

Se il test ha esito positivo, questo livello mostra lo stato di output e il risultato dell'elaborazione dei dati di input e output.

Se il test fallisce, questo livello mostra l'output dell'errore. Questo livello mostra i risultati intermedi dell'elaborazione dei dati fino al punto di errore. Ad esempio, supponiamo di aver testato uno stato Task che richiama una funzione. Lambda Immagina di aver applicato i OutputPath filtriInputPath, ParametriResultPath, e allo stato Task. Supponiamo che l'invocazione sia fallita. In questo caso, il

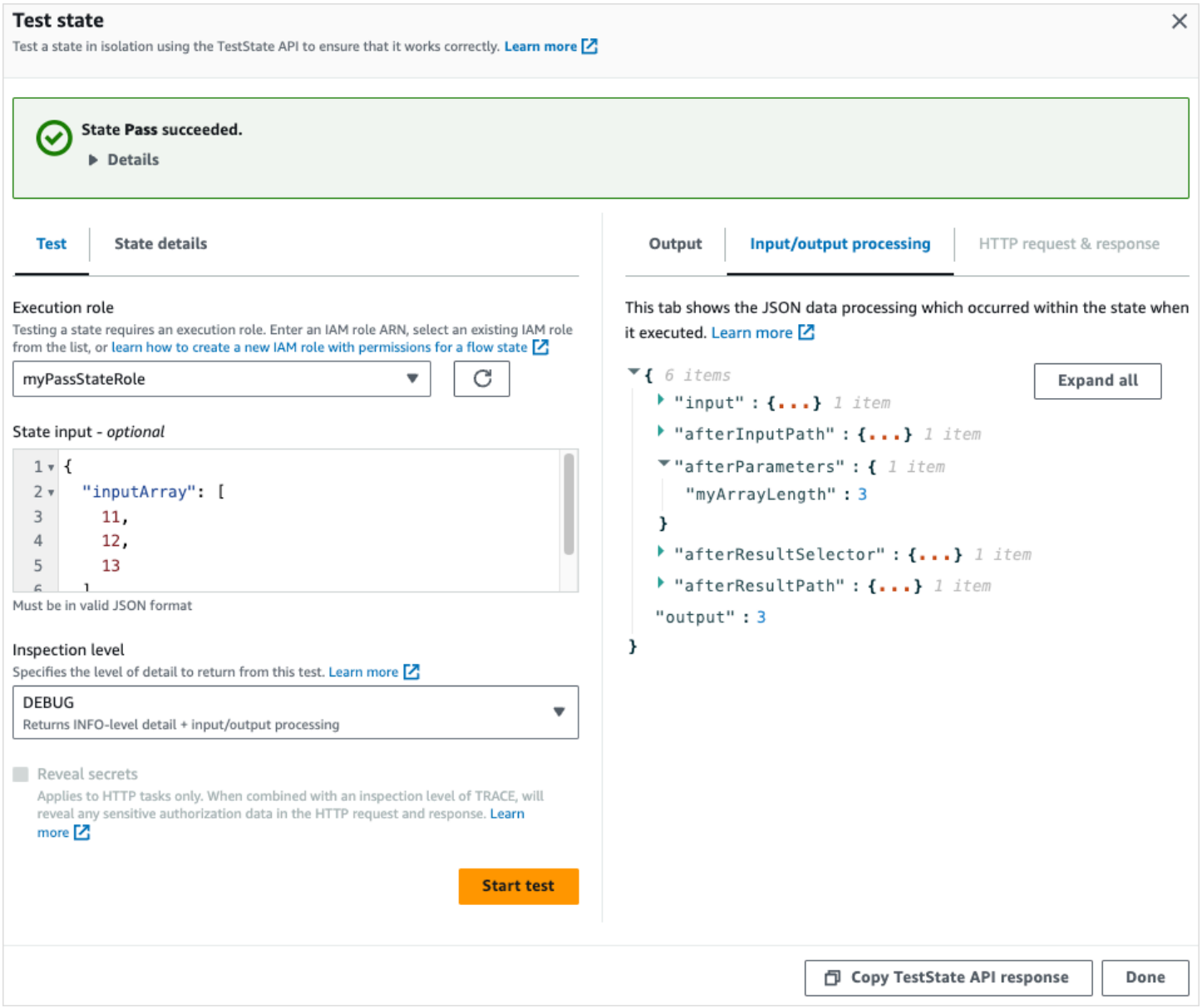
DEBUG livello mostra i risultati dell'elaborazione dei dati in base all'applicazione dei filtri nell'ordine seguente:

- `input`— Input in stato grezzo
- `afterInputPath`— Input after Step Functions applica il `InputPath` filtro.
- `afterParameters`— L'input effettivo dopo l'Step Functionsapplicazione del `Parameters` filtro.

Le informazioni diagnostiche disponibili in questo livello possono aiutarti a risolvere i problemi relativi all'integrazione di un servizio o al flusso di elaborazione dei dati di input e output che potresti aver definito.

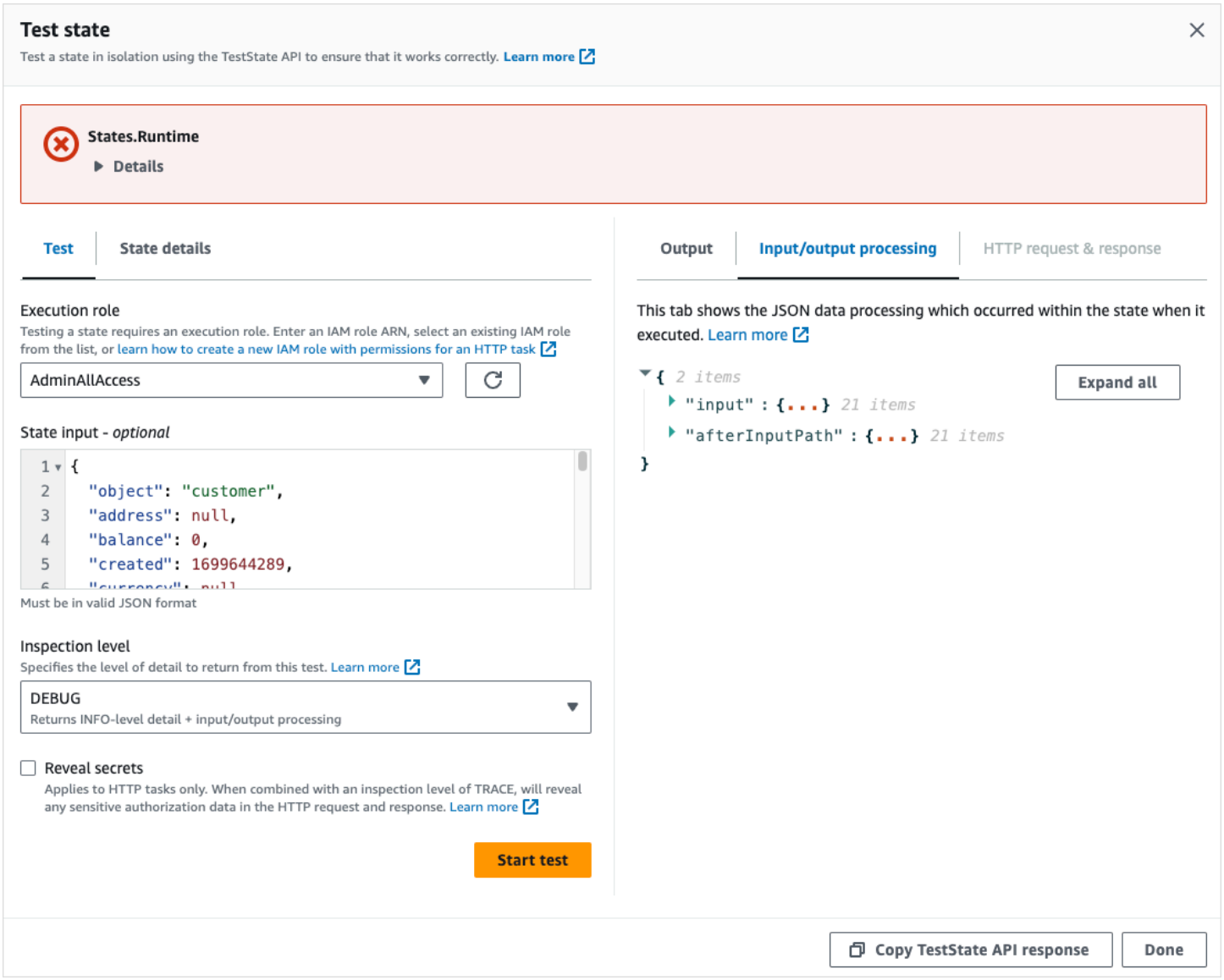
Esempio di test con livello DEBUG che ha esito positivo

L'immagine seguente mostra un test per uno stato Pass che ha esito positivo. Il livello di ispezione per questo stato è impostato su DEBUG. La scheda Elaborazione Input/Output nell'immagine seguente mostra il risultato dell'applicazione di `Parameters`sull'input fornito per questo stato.



Esempio di test con livello DEBUG che fallisce

L'immagine seguente mostra un test fallito per uno stato Task quando il livello di ispezione è impostato su DEBUG. La scheda Elaborazione input/output nell'immagine seguente mostra il risultato dell'elaborazione dei dati di input e output per lo stato fino al punto in cui si è verificato il fallimento.



Livello di ispezione TRACE

Step Functionsfornisce il livello TRACE per testare un task HTTP. Questo livello restituisce informazioni sulla richiesta HTTP che Step Functions effettua e sulla risposta restituita da un'API di terze parti. La risposta potrebbe contenere informazioni, come le intestazioni e il corpo della richiesta. Inoltre, è possibile visualizzare l'output dello stato e il risultato dell'elaborazione dei dati di input e output in questo livello.

Se il test fallisce, questo livello mostra l'output dell'errore.

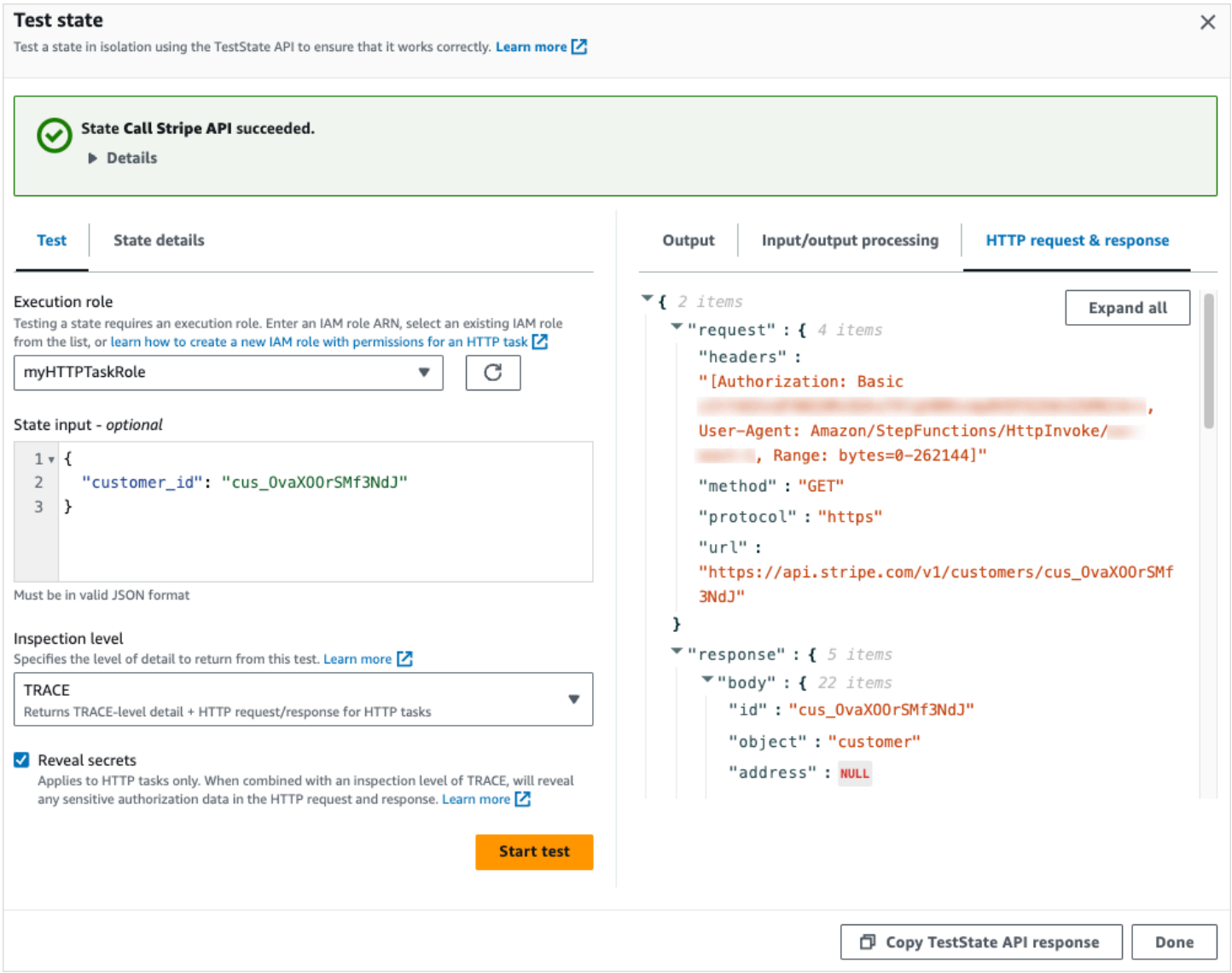
Questo livello è applicabile solo per HTTP Task. Step Functionsgenera un errore se si utilizza questo livello per altri tipi di stato.

Quando si imposta il livello di ispezione su TRACE, è possibile visualizzare anche i segreti inclusi nella EventBridge connessione. A tale scopo, è necessario impostare il `revealSecrets` parametro su `true` nell'TestStateAPI. Inoltre, devi assicurarti che l'IAMutente che chiama l' TestState API disponga dell'autorizzazione per l'`states:RevealSecrets`azione. Per un esempio di IAM politica che imposta l'`states:RevealSecrets`autorizzazione, vediIAMautorizzazioni per l'utilizzo dell'API TestState . Senza questa autorizzazione, Step Functions genera un errore di accesso negato.

Se si imposta il `revealSecrets` parametro su`false`, Step Functions omette tutti i segreti nei dati di richiesta e risposta HTTP.

Esempio di test con livello TRACE che ha esito positivo

L'immagine seguente mostra un test per un task HTTP che ha esito positivo. Il livello di ispezione per questo stato è impostato su TRACE. La scheda Richiesta e risposta HTTP nell'immagine seguente mostra il risultato della chiamata API di terze parti.



IAMautorizzazioni per l'utilizzo dell'API TestState

L'IAMutente che chiama l'`TestState`API deve disporre delle autorizzazioni per eseguire le azioni `states:TestState` and`iam:PassRole`. Inoltre, se si imposta il parametro RevealSecrets su`true`, è necessario assicurarsi che l'IAMutente disponga delle autorizzazioni per eseguire l'azione. `states:RevealSecrets` Senza questa autorizzazione, Step Functions genera un errore di accesso negato.

È inoltre necessario assicurarsi che il ruolo di esecuzione contenga le IAM autorizzazioni necessarie per le risorse a cui accede lo stato. Per informazioni sulle autorizzazioni di cui uno stato potrebbe aver bisogno, consulta Gestione dei ruoli di esecuzione.

Il seguente esempio di IAM policy imposta le `states:RevealSecrets` autorizzazioni `states:TestStateiam:PassRole`, e.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:TestState",
        "states:RevealSecrets",
        "iam:PassRole"
      ],
      "Resource": "*"
    }
  ]
}
```

Test di uno stato (console)

È possibile testare uno stato nella console e controllare lo stato di output o il flusso di elaborazione dei dati di input e output. Per un task HTTP, puoi testare la richiesta e la risposta HTTP non elaborate.

Per testare uno stato

1. Apri la console Step Functions.
2. Scegli Crea macchina a stati per iniziare a creare una macchina a stati o scegli una macchina a stati esistente.
3. In Workflow Studio, scegli lo stato che desideri testare. Modalità di progettazione
4. Scegliete Stato di test nel ✓Inspector pannello di Workflow Studio.
5. Nella finestra di dialogo Stato di test, effettuate le seguenti operazioni:

    a. Per Ruolo di esecuzione, scegliete un ruolo di esecuzione per testare lo stato. Assicurati di disporre delle IAMautorizzazioni necessarie per lo stato che desideri testare.

    b. (Facoltativo) Fornisci qualsiasi input JSON necessario allo stato selezionato per il test.

    c. Per Livello di ispezione, selezionate una delle seguenti opzioni in base ai valori che desiderate visualizzare:

- INFO: mostra l'output dello stato nella scheda Output se il test ha esito positivo. Se il test fallisce, INFO mostra l'output dell'errore, che include il nome dell'errore e una spiegazione dettagliata della causa dell'errore. Per impostazione predefinita, Step Functions imposta il livello di ispezione su INFO se non si seleziona un livello.
- DEBUG: mostra l'output dello stato e il risultato dell'elaborazione dei dati di input e output se il test ha esito positivo. Se il test fallisce, DEBUG mostra l'output dell'errore, che include il nome dell'errore e una spiegazione dettagliata della causa dell'errore.
- TRACE: mostra la richiesta e la risposta HTTP non elaborate ed è utile per verificare intestazioni, parametri di query e altri dettagli specifici dell'API. Questa opzione è disponibile solo per il task HTTP.

  Facoltativamente, puoi scegliere di selezionare Rivela segreti. In combinazione con TRACE, questa impostazione consente di visualizzare i dati sensibili inseriti dalla EventBridge connessione, come le chiavi API. L'identità IAM utente che utilizzi per accedere alla console deve disporre dell'autorizzazione per eseguire l'`states:RevealSecrets`azione. Senza questa autorizzazione, all'avvio Step Functions del test viene generato un errore di accesso negato. Per un esempio di IAM policy che imposta l'`states:RevealSecrets`autorizzazione, vediIAMautorizzazioni per l'utilizzo dell'API TestState .

d. Scegli Avvia test.

Testare uno stato utilizzando AWS CLI

È possibile testare uno stato supportato utilizzando l'TestStateAPI inAWS CLI. Questa API accetta la definizione di uno stato e la esegue.

Per ogni stato, puoi specificare la quantità di dettagli che desideri visualizzare nei risultati del test. Questi dettagli forniscono informazioni aggiuntive sull'esecuzione dello stato, inclusi i risultati dell'elaborazione dei dati di input e output e le informazioni su richieste e risposte HTTP. Gli esempi seguenti mostrano i diversi livelli di ispezione che è possibile specificare per l' TestState API. Ricordatevi di sostituire il testo in *`corsivo`* con le informazioni specifiche della risorsa.

Questa sezione contiene i seguenti esempi che descrivono come utilizzare i diversi livelli di ispezione disponibili in: Step Functions AWS CLI

- Utilizzo di INFO InspectionLevel

- Utilizzo di DEBUG InspectionLevel
- Utilizzo di TRACE InspectionLevel
- Utilizzo dell'utilità jq AWS CLI per filtrare e stampare la risposta HTTP restituita dall'API TestState

Esempio 1: utilizzo di INFO InspectionLevel per testare uno stato Choice

Per testare uno stato utilizzando `INFO` InspectionLevel in AWS CLI, esegui il `test-state` comando come mostrato nell'esempio seguente.

```
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}'
```

Questo esempio utilizza uno stato Choice per determinare il percorso di esecuzione dello stato in base all'input numerico fornito. Per impostazione predefinita, Step Functions imposta `inspectionLevel` su `INFO` se non si imposta un livello.

Step Functionsrestituisce il seguente risultato.

```
{
    "output": "{\"number\": 2}",
    "nextState": "Equals 2",
    "status": "SUCCEEDED"
}
```

Esempio 2: utilizzo di DEBUG inspectionLevel per eseguire il debug dell'elaborazione dei dati di input e output in uno stato Pass

Per testare uno stato utilizzando `DEBUG` InspectionLevel in AWS CLI, esegui il comando come mostrato nell'esempio seguente`test-state`.

```
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
```

Questo esempio utilizza uno Pass stato per mostrare come Step Functions filtra e manipola i dati JSON di input utilizzando i filtri di elaborazione dei dati di input e output. Questo esempio utilizza i seguenti filtri:`InputPath`,, e`Parametri`. `ResultPath` `OutputPath`

Step Functionsrestituisce il seguente risultato.

```
{
    "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"
}
```

Esempio 3: utilizzo di TRACE InspectionLevel e RevealSecrets per ispezionare la richiesta HTTP inviata a un'API di terze parti

Per testare un task HTTP utilizzando TRACE InspectionLevel insieme al parametro RevealSecrets in, esegui il comando come illustrato nell' AWS CLIesempio seguente. `test-state`

```
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
```

Questo esempio verifica se il task HTTP chiama l'API di terze parti specificata,. `https://httpbin.org/` Mostra anche i dati di richiesta e risposta HTTP per la chiamata 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"
```

```
}
```

Esempio 4: utilizzo dell'utilità jq per filtrare e stampare la risposta restituita dall' TestState API

L' TestState API restituisce i dati JSON come stringhe di escape nella sua risposta. L'AWS CLIesempio seguente estende l'esempio 3 e utilizza l'`jq`utilità per filtrare e stampare la risposta HTTP restituita dall' TestState API in un formato leggibile dall'uomo. Per informazioni `jq` e istruzioni di installazione, consulta 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'
```

L'esempio seguente mostra l'output restituito in un formato leggibile dall'uomo.

```
{
  "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"
}
```

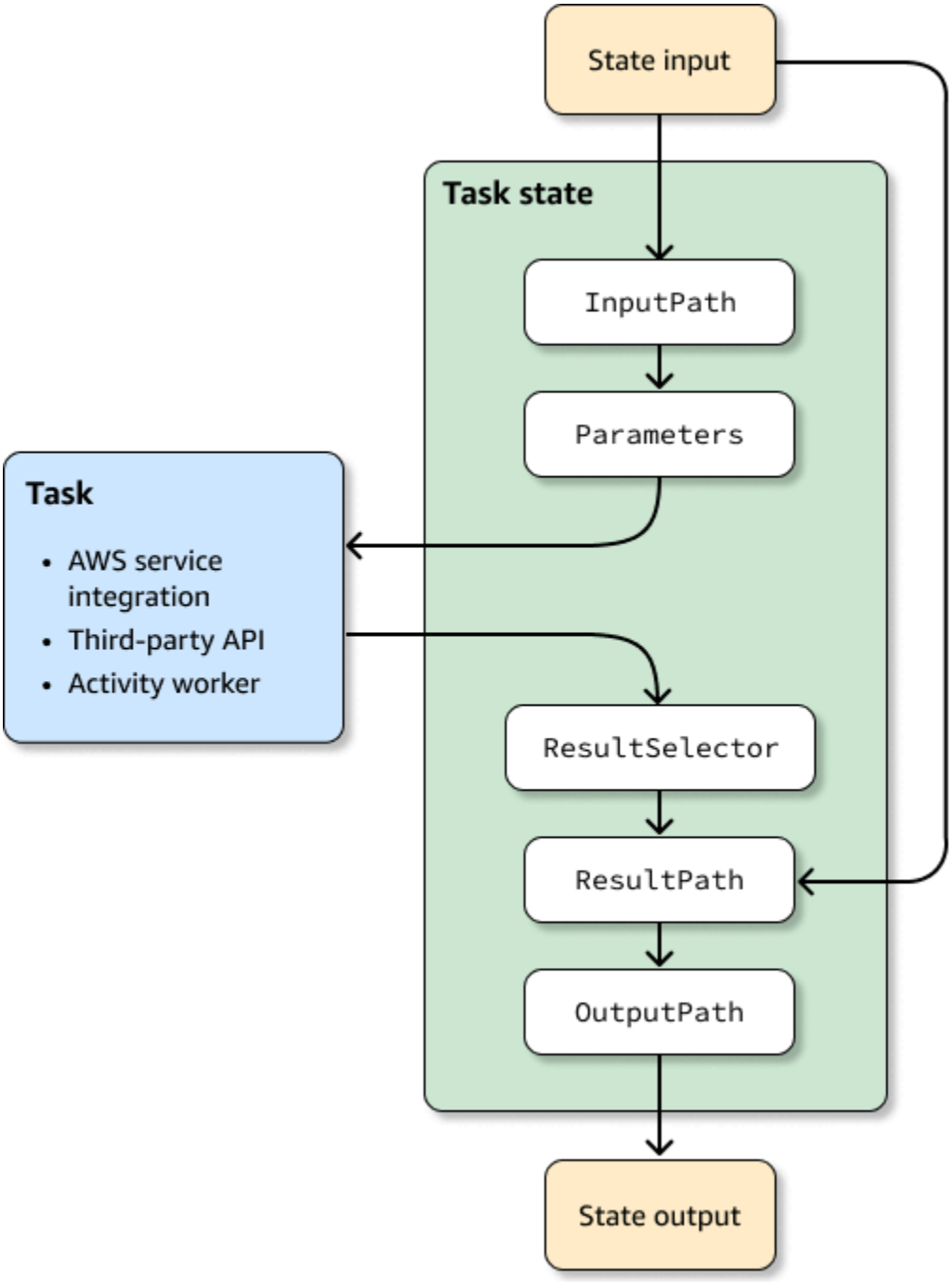
Test e debug del flusso di dati di input e output

L'`TestState`API è utile per testare ed eseguire il debug dei dati che fluiscono attraverso il flusso di lavoro. Questa sezione fornisce alcuni concetti chiave e spiega come utilizzarli TestState per questo scopo.

Concetti chiave

NelStep Functions, il processo di filtraggio e manipolazione dei dati JSON mentre attraversano gli stati della macchina a stati viene chiamato elaborazione di input e output. Per ulteriori informazioni sul funzionamento, consulta Elaborazione di input e output in Step Functions.

Tutti i tipi di stato presenti in Amazon States Language (ASL) (Task, Parallel, Map, Pass, Wait, Choice, Succeed e Fail) condividono una serie di campi comuni per filtrare e manipolare i dati JSON che li attraversano. Questi campi sono:InputPath,,, eParametri. ResultSelector ResultPath OutputPath Il supporto per ogni campo varia a seconda degli stati. In fase di esecuzione, Step Functions applica ogni campo in un ordine specifico. Il diagramma seguente mostra l'ordine in cui questi campi vengono applicati ai dati all'interno di uno stato del task:



L'elenco seguente descrive l'ordine di applicazione dei campi di elaborazione di input e output mostrati nel diagramma.

1. L'input di stato è costituito dai dati JSON passati allo stato corrente da uno stato precedente.
2. InputPathfiltra una parte dell'input allo stato grezzo.
3. Parametriconfigura l'insieme di valori da passare al Task.
4. L'attività esegue un lavoro e restituisce un risultato.

5. ResultSelectorseleziona un insieme di valori da escludere dal risultato dell'attività.
6. ResultPathcombina il risultato con l'input allo stato grezzo o lo sostituisce con esso.
7. OutputPathfiltra una parte dell'output per farla passare allo stato successivo.
8. Lo stato di output è costituito dai dati JSON passati dallo stato corrente allo stato successivo.

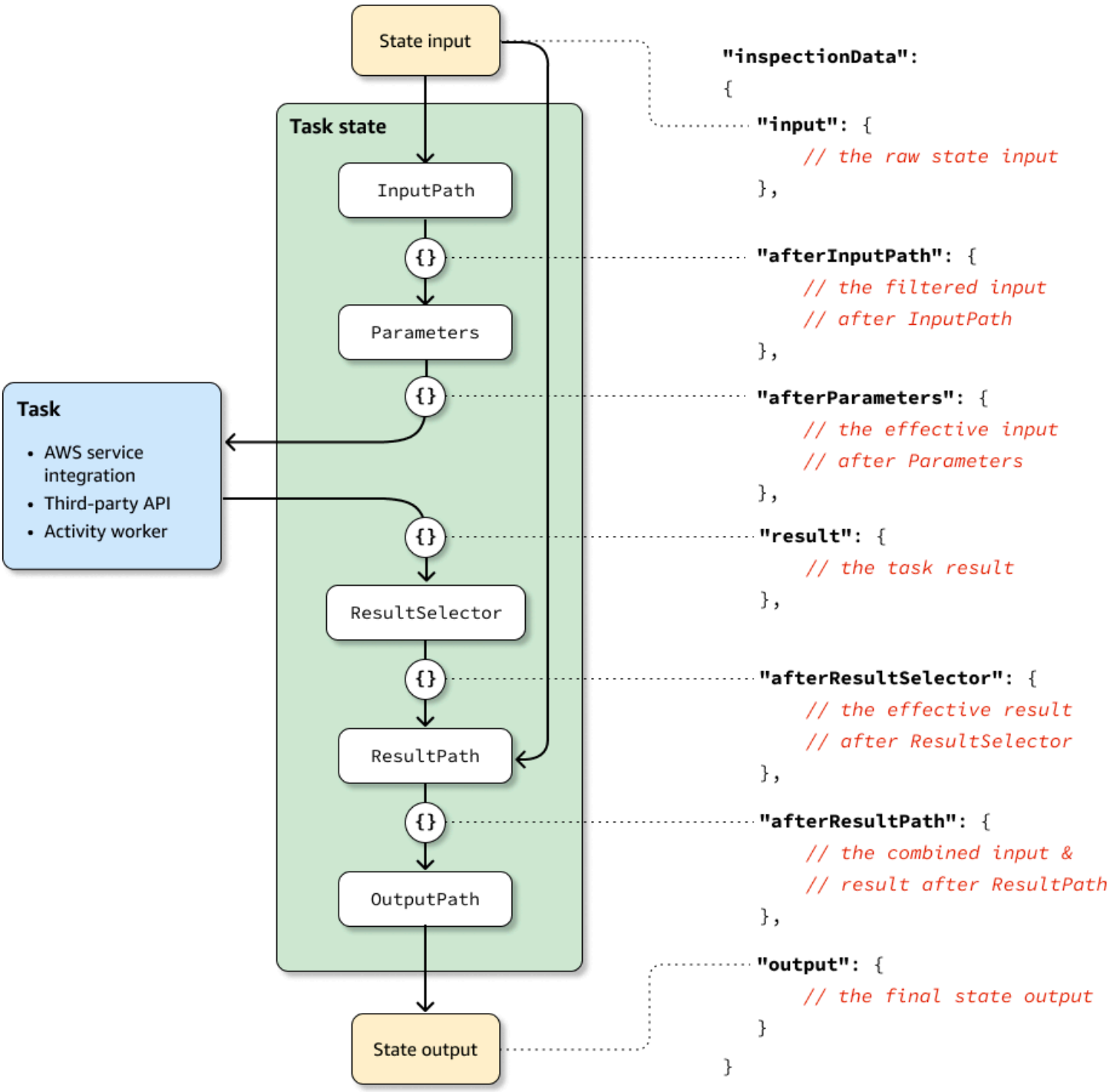
Questi campi di elaborazione di input e output sono opzionali. Se non si utilizza nessuno di questi campi nella definizione dello stato, l'attività utilizzerà l'input dello stato non elaborato e restituirà il risultato dell'attività come output dello stato.

Utilizzo TestState per ispezionare l'elaborazione di input e output

Quando si chiama l'`TestState`API e si imposta il `inspectionLevel` parametro su`DEBUG`, la risposta dell'API include un oggetto chiamato`inspectionData`. Questo oggetto contiene campi che consentono di verificare in che modo i dati sono stati filtrati o manipolati all'interno dello stato in cui sono stati eseguiti. L'esempio seguente mostra l'`inspectionData`oggetto per uno stato Task.

```
"inspectionData":   {
  "input": string,
  "afterInputPath": string,
  "afterParameters": string,
  "result": string,
  "afterResultSelector": string,
  "afterResultPath": string,
  "output": string
}
```

In questo esempio, ogni campo che contiene il `after` prefisso mostra i dati dopo l'applicazione di un determinato campo. Ad esempio, `afterInputPath` mostra l'effetto dell'applicazione del `InputPath` campo per filtrare l'input allo stato grezzo. Il diagramma seguente mappa ogni campo di definizione ASL al campo corrispondente nell'`inspectionData`oggetto:



Per esempi di utilizzo dell' TestState API per il debug dell'elaborazione di input e output, consulta quanto segue:

- Test di uno stato utilizzando il livello di ispezione DEBUG nella console Step Functions
- Test di uno stato utilizzando il livello di ispezione DEBUG in AWS CLI

Test di macchine a stato locale

AWSStep Functions Local è una versione scaricabile di Step Functions che consente di sviluppare e testare applicazioni utilizzando una versione di Step Functions in esecuzione nel proprio ambiente di sviluppo. La versione locale di Step Functions può richiamare AWS Lambda funzioni, sia durante l'esecuzione locale che durante l'esecuzione. AWS È inoltre possibile coordinare altri servizi AWS supportati.

> **Note**
>
> Step Functions Local utilizza account fittizi per funzionare.

Durante l'esecuzione di Step Functions Local, puoi utilizzare uno dei seguenti modi per richiamare le integrazioni dei servizi:

- Configurazione degli endpoint locali per AWS Lambda e altri servizi. Per informazioni sugli endpoint supportati, consulta. Impostazione delle opzioni di configurazione per Step Functions Local
- Effettuare chiamate direttamente a un AWS servizio da Step Functions Local.
- Prendere in giro la risposta delle integrazioni di servizi. Per informazioni sull'utilizzo di integrazioni di servizi simulate, consulta. Utilizzo di integrazioni di servizi simulati

AWSStep Functions Local è disponibile come pacchetto JAR o immagine Docker autonoma che può essere eseguita su Microsoft Windows, Linux, macOS e altre piattaforme che supportano Java o Docker.

> **Warning**
>
> La versione scaricabile di AWS Step Functions è pensata per essere utilizzata solo per i test e non deve mai essere utilizzata per elaborare informazioni sensibili.

> **Tip**
>
> Assicurati di utilizzare Step Functions Local versione 1.12.0 o successiva per poter includere tutte le funzioni intrinseche nei tuoi flussi di lavoro.

I seguenti argomenti descrivono come configurare Step Functions Local utilizzando Docker e il file JAR ed eseguire Step Functions Local per lavorare conAWS Lambda, AWS Serverless Application Model (AWS SAM) CLI Local o altri servizi supportati.

Argomenti

- Configurazione di Step Functions Local (versione scaricabile) e Docker
- Configurazione locale di Step Functions (versione scaricabile) - Versione Java
- Impostazione delle opzioni di configurazione per Step Functions Local
- Esecuzione di Step Functions Local sul computer
- Test di Step Functions e AWS SAM CLI Local
- Utilizzo di integrazioni di servizi simulati

Configurazione di Step Functions Local (versione scaricabile) e Docker

L'immagine Docker locale di Step Functions consente di iniziare rapidamente con Step Functions Local utilizzando un'immagine Docker con tutte le dipendenze necessarie. L'immagine Docker consente di includere Step Functions Local nelle build containerizzate e come parte dei test di integrazione continua.

Per ottenere l'immagine Docker per Step Functions Local, vedi https://hub.docker.com/r/amazon/aws-stepfunctions-local o inserisci il seguente comando Docker`pull`.

```
docker pull amazon/aws-stepfunctions-local
```

Per avviare la versione scaricabile di Step Functions on Docker, esegui il seguente comando Docker `run`

```
docker run -p 8083:8083 amazon/aws-stepfunctions-local
```

Per interagire con AWS Lambda o con altri servizi supportati, devi prima configurare le tue credenziali e altre opzioni di configurazione. Per ulteriori informazioni, consulta i seguenti argomenti:

- Impostazione delle opzioni di configurazione per Step Functions Local
- Credenziali e configurazione per Docker

Configurazione locale di Step Functions (versione scaricabile) - Versione Java

La versione scaricabile di AWS Step Functions viene fornita come file JAR eseguibile e come immagine Docker. L'applicazione Java è eseguibile su Windows, Linux, macOS e altre piattaforme compatibili con Java. Oltre Java, occorre installare AWS Command Line Interface (AWS CLI). Per informazioni sull'installazione e la configurazione diAWS CLI, consulta la Guida per l'utente. AWS Command Line Interface

Per configurare ed eseguire Step Functions sul tuo computer

1. Scaricate Step Functions utilizzando i seguenti link.

Collegamenti per il download	Checksum
.tar.gz	.tar.gz.md5
.zip	.zip.md5

2. Estrarre il file `.zip`.
3. Test del download e visualizzazione delle informazioni sulla versione.

```
$ java -jar StepFunctionsLocal.jar -v
Step Function Local
Version: 1.0.0
Build: 2019-01-21
```

4. (Facoltativo) Visualizzare un elenco dei comandi disponibili.

```
$ java -jar StepFunctionsLocal.jar -h
```

5. Per avviare Step Functions sul tuo computer, apri un prompt dei comandi, vai alla directory in cui hai estratto `StepFunctionsLocal.jar` e immetti il comando seguente.

```
java -jar StepFunctionsLocal.jar
```

6. Per accedere a Step Functions in esecuzione localmente, utilizzare il `--endpoint-url` parametro. Ad esempio, utilizzandoAWS CLI, è possibile specificare i comandi Step Functions come segue:

```
aws stepfunctions --endpoint-url http://localhost:8083 command
```

> **Note**
>
> Per impostazione predefinita, Step Functions Local utilizza un account di test e credenziali locali e la AWS regione è impostata su Stati Uniti orientali (Virginia settentrionale). Per utilizzare Step Functions Local con AWS Lambda o altri servizi supportati, è necessario configurare le credenziali e la regione.
> Se si utilizzano flussi di lavoro Express con Step Functions Local, la cronologia di esecuzione verrà archiviata in un file di registro. Non viene registrata in Logs. CloudWatch Il percorso del file di registro sarà basato sull'ARN del gruppo di log CloudWatch Logs fornito quando si crea la macchina a stati locale. Il file di registro verrà memorizzato in `/aws/states/log-group-name/${execution_arn}.log` relazione alla posizione in cui è in esecuzione Step Functions Local. Ad esempio, se l'ARN di esecuzione è:
>
> ```
> arn:aws:states:us-east-1:123456789012:express:test:example-ExpressLogGroup-wJalrXUtnFEMI
> ```
>
> il file di registro sarà:
>
> ```
> aws/states/log-group-name/arn:aws:states:us-east-1:123456789012:express:test:example-ExpressLogGroup-wJalrXUtnFEMI.log
> ```

Impostazione delle opzioni di configurazione per Step Functions Local

Quando avviate AWS Step Functions Local utilizzando il file JAR, potete impostare le opzioni di configurazione utilizzando AWS Command Line Interface (AWS CLI) o includendole nell'ambiente di sistema. Per Docker, è necessario specificare queste opzioni in un file a cui si fa riferimento all'avvio di Step Functions Local.

Opzioni di configurazione

Quando configuri il contenitore Step Functions Local per utilizzare un endpoint di override come Lambda Endpoint e Batch Endpoint ed effettui chiamate a tale endpoint, Step Functions Local non utilizza le credenziali specificate. L'impostazione di queste sostituzioni degli endpoint è facoltativa.

Opzione	Riga di comando	Ambiente
Account	-account, --aws-account	AWS_ACCOUNT_ID
Region	-regione, --aws-region	AWS_DEFAULT_REGION
Wait Time Scale	-waitTimeScale, --wait-time-scale	WAIT_TIME_SCALE
Endpoint Lambda	-LambdaEndpoint, --lambda-endpoint	LAMBDA_ENDPOINT
Endpoint Batch	-batchEndpoint, --batch-endpoint	BATCH_ENDPOINT
Endpoint DynamoDB	-DynamoDB Endpoint, --dynamodb-endpoint	DYNAMODB_ENDPOINT
Endpoint ECS	-ecsEndpoint, --ecs-endpoint	ECS_ENDPOINT
Endpoint Glue	-glueEndpoint, --glue-endpoint	GLUE_ENDPOINT
SageMaker Punto finale	-sageMakerEndpoint, --sagemaker-endpoint	SAGE_MAKER_ENDPOINT
Endpoint SQS	-sqsEndpoint, --sqs-endpoint	SQS_ENDPOINT
Endpoint SNS	-snsEndpoint, --sns-endpoint	SNS_ENDPOINT
Endpoint Step Functions	-stepFunctionsEndpoint, --step-functions-endpoint	STEP_FUNCTIONS_ENDPOINT

Credenziali e configurazione per Docker

Per configurare Step Functions Local for Docker, crea il seguente file:`aws-stepfunctions-local-credentials.txt`.

Questo file contiene le tue credenziali e altre opzioni di configurazione. Quanto segue può essere usato come modello durante la creazione del `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
```

Dopo aver configurato le credenziali e le opzioni di configurazione in`aws-stepfunctions-local-credentials.txt`, avvia Step Functions con il seguente comando.

```
docker run -p 8083:8083 --env-file aws-stepfunctions-local-credentials.txt amazon/aws-stepfunctions-local
```

> **Note**
>
> Si consiglia di utilizzare il nome DNS specialehost.docker.internal, che si risolve nell'indirizzo IP interno utilizzato dall'host, ad esempio. `http://host.docker.internal:8000` Per ulteriori informazioni, consulta la documentazione Docker per Mac e Windows rispettivamente alle funzionalità di rete in Docker Desktop per Mac e alle funzionalità di rete in Docker Desktop per Windows.

Esecuzione di Step Functions Local sul computer

Usa la versione locale di Step Functions per configurare, sviluppare e testare macchine a stati sul tuo computer.

Esegui una macchina a HelloWorld stati localmente

Dopo aver eseguito Step Functions localmente con AWS Command Line Interface (AWS CLI), è possibile avviare l'esecuzione di una macchina a stati.

1. Crea una macchina a stati da AWS CLI escludendo la definizione della macchina a stati.

```
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**
>
> Non `role-arn` viene utilizzato per Step Functions Local, ma è necessario includerlo con la sintassi corretta. È possibile utilizzare l'Amazon Resource Name (ARN) dall'esempio precedente.

Se la macchina a stati è stata creata correttamente, Step Functions risponde con la data di creazione e l'ARN della macchina a stati.

```
{
    "creationDate": 1548454198.202,
    "stateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:HelloWorld"
}
```

2. Avvia un'esecuzione utilizzando l'ARN della macchina a stati creata.

```
aws stepfunctions --endpoint-url http://localhost:8083 start-execution --state-
machine-arn arn:aws:states:us-east-1:123456789012:stateMachine:HelloWorld
```

Step Functions Local con AWS SAM CLI Local

È possibile utilizzare la versione locale di Step Functions con una versione locale diAWS Lambda. Per la configurazione, è necessario installare e configurare AWS SAM.

Per informazioni sulla configurazione e l'esecuzione di AWS SAM, consultare:

- Configurazione di AWS SAM
- Avviare AWS SAM CLI Local.

Quando Lambda è in esecuzione sul tuo sistema locale, puoi avviare Step Functions Local. Dalla directory in cui hai estratto i file JAR locali di Step Functions, avvia Step Functions Local e usa il `--lambda-endpoint` parametro per configurare l'endpoint Lambda locale.

```
java -jar StepFunctionsLocal.jar --lambda-endpoint http://127.0.0.1:3001 command
```

Per ulteriori informazioni sull'esecuzione di Step Functions Local conAWS Lambda, vedereTest di Step Functions e AWS SAM CLI Local.

Test di Step Functions e AWS SAM CLI Local

Con entrambi AWS Step Functions e AWS Lambda in esecuzione sul computer locale, puoi testare la tua State Machine e le funzioni Lambda senza distribuire il tuo codice su. AWS

Per ulteriori informazioni, consulta i seguenti argomenti:

- Test di macchine a stato locale
- Configurazione di AWS SAM

Argomenti

- Fase 1: configurare AWS SAM
- Fase 2: esecuzione dei test per AWS SAM CLI Local
- Fase 3: avvio di AWS SAM CLI Local
- Fase 4: Avvia Step Functions Local
- Fase 5: creazione di una macchina a stati che faccia riferimento a una funzione AWS SAM CLI Local
- Fase 6: avviare un'esecuzione della macchina a stati locale.

Fase 1: configurare AWS SAM

AWS Serverless Application Model (AWS SAM) CLI Local richiede l'installazione di AWS Command Line Interface, AWS SAM e Docker.

1. Installazione della CLI AWS SAM.

   > **Note**
   >
   > Prima di installare la CLI AWS SAM, è necessario installare AWS CLI e Docker. Vedi Prerequisiti per l'installazione dell'interfaccia a riga di comando AWS SAM.

2. Vai alla documentazione Quick Start AWS SAM. Assicurati di seguire i passaggi per:

   1. Inizializzare l'applicazione
   2. Testare l'applicazione in locale

   Questo crea una `sam-app` directory e crea un ambiente che include una funzione Hello World Lambda basata su Python.

Fase 2: esecuzione dei test per AWS SAM CLI Local

Ora che hai installato AWS SAM e creato la funzione Hello World Lambda, puoi testarla. Nella directory `sam-app`, immettere il seguente comando:

```
sam local start-api
```

Viene avviata un'istanza locale della funzione Lambda. Dovresti vedere un output simile al seguente:

```
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)
```

Apri un browser e inserisci quanto segue:

```
http://127.0.0.1:3000/hello
```

Questo produrrà una risposta simile alla seguente:

```
{"message": "hello world", "location": "72.21.198.66"}
```

Inserisci CTRL+C per terminare l'API Lambda.

Fase 3: avvio di AWS SAM CLI Local

Dopo avere testato il corretto funzionamento della funzione, avviare AWS SAM CLI Local. Nella directory `sam-app`, immettere il seguente comando:

```
sam local start-lambda
```

Questo avvia AWS SAM CLI Local e fornisce l'endpoint da utilizzare, simile al seguente output:

```
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)
```

Fase 4: Avvia Step Functions Local

File JAR

Se utilizzi la versione `.jar` file di Step Functions Local, avvia Step Functions e specifica l'endpoint Lambda. Nella directory in cui hai estratto i `.jar` file, inserisci il seguente comando:

```
java -jar StepFunctionsLocal.jar --lambda-endpoint http://localhost:3001
```

All'avvio di Step Functions Local, verifica l'ambiente e quindi le credenziali configurate nel file`~/.aws/credentials`. Per impostazione predefinita, inizia a utilizzare un ID utente fittizio ed è elencato come. `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

Se stai utilizzando la versione Docker di Step Functions Local, avvia Step Functions con il seguente comando:

```
docker run -p 8083:8083 amazon/aws-stepfunctions-local
```

Per informazioni sull'installazione della versione Docker di Step Functions, vedereConfigurazione di Step Functions Local (versione scaricabile) e Docker.

> **Note**
>
> È possibile specificare l'endpoint tramite la riga di comando o impostando le variabili di ambiente se si avvia Step Functions dal `.jar` file. Per la versione Docker, è necessario specificare gli endpoint e le credenziali in un file di testo. Consultare Impostazione delle opzioni di configurazione per Step Functions Local.

Fase 5: creazione di una macchina a stati che faccia riferimento a una funzione AWS SAM CLI Local

Una volta che Step Functions Local è in esecuzione, crea una macchina a stati che faccia riferimento a `HelloWorldFunction` quella in Fase 1: configurare AWS SAM cui hai inizializzato.

```
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"
```

Questo creerà una macchina a stati e fornirà un Amazon Resource Name (ARN) che puoi utilizzare per avviare un'esecuzione.

```
{
    "creationDate": 1548805711.403,
    "stateMachineArn": "arn:aws:states:us-east-1:123456789012:stateMachine:HelloWorld"
}
```

Fase 6: avviare un'esecuzione della macchina a stati locale.

Dopo aver creato una macchina a stati, avvia un'esecuzione. Dovrai fare riferimento all'endpoint e all'ARN della macchina a stati quando usi il seguente comando: **`aws stepfunctions`**

```
aws stepfunctions --endpoint http://localhost:8083 start-execution --state-machine
 arn:aws:states:us-east-1:123456789012:stateMachine:HelloWorld --name test
```

Questo avvia un'esecuzione denominata `test` della tua macchina a `HelloWorld` stati.

```
{
    "startDate": 1548810641.52,
    "executionArn": "arn:aws:states:us-east-1:123456789012:execution:HelloWorld:test"
}
```

Ora che Step Functions è in esecuzione localmente, puoi interagire con esso utilizzando ilAWS CLI. Ad esempio, per ottenere informazioni su questa esecuzione, utilizzare il seguente comando:

```
aws stepfunctions --endpoint http://localhost:8083 describe-execution --execution-arn
 arn:aws:states:us-east-1:123456789012:execution:HelloWorld:test
```

La `describe-execution` richiesta di esecuzione fornisce dettagli più completi, simili al seguente output:

```
{
    "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": "{}"
}
```

Utilizzo di integrazioni di servizi simulati

In Step Functions Local, puoi testare i percorsi di esecuzione delle tue macchine a stati senza chiamare effettivamente servizi integrati utilizzando integrazioni di servizi simulate. Per configurare le macchine a stati per utilizzare integrazioni di servizi simulate, create un file di configurazione fittizio. In questo file, definisci l'output desiderato delle integrazioni di servizio come risposte simulate e le esecuzioni che utilizzano le risposte simulate per simulare un percorso di esecuzione come casi di test.

Fornendo il file di configurazione fittizio a Step Functions Local, è possibile testare le chiamate di integrazione dei servizi eseguendo macchine a stati che utilizzano le risposte simulate specificate nei casi di test invece di effettuare chiamate di integrazione del servizio effettive.

> **Note**
>
> Se non specifichi risposte simulate di integrazione del servizio nel file di configurazione fittizio, Step Functions Local invocherà l'integrazione del AWS servizio utilizzando l'endpoint configurato durante la configurazione di Step Functions Local. Per informazioni sulla configurazione degli endpoint per Step Functions Local, vedere. Impostazione delle opzioni di configurazione per Step Functions Local

Argomenti

- Concetti chiave in questo argomento
- Passaggio 1: specificare le integrazioni di servizi simulati in un file di configurazione fittizio
- Passaggio 2: fornire il file di configurazione fittizio a Step Functions Local
- Fase 3: Eseguire i test di integrazione dei servizi simulati
- File di configurazione per integrazioni di servizi fittizi

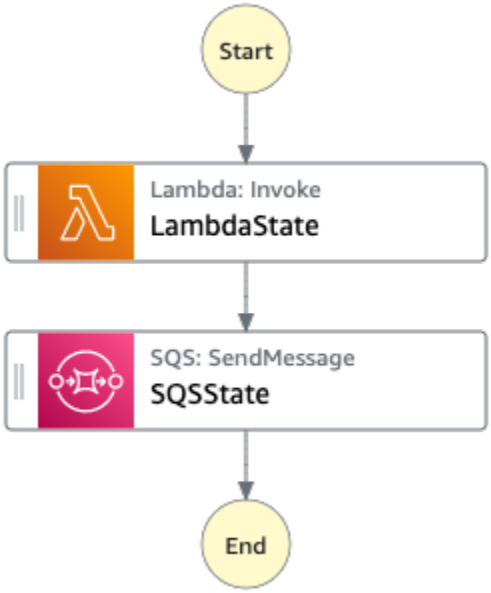
Concetti chiave in questo argomento

In questo argomento vengono utilizzati diversi concetti definiti nell'elenco seguente:

- Integrazioni di servizi simulati: si riferisce agli stati delle attività configurati per utilizzare risposte simulate anziché eseguire chiamate di servizio effettive.
- Risposte simulate: si riferisce a dati fittizi per i quali gli stati di Task possono essere configurati per l'utilizzo.
- Casi di test: si riferisce alle esecuzioni di macchine a stati configurate per utilizzare integrazioni di servizi simulate.
- File di configurazione fittizio: si riferisce al file di configurazione fittizio che contiene JSON, che definisce integrazioni di servizi simulate, risposte simulate e casi di test.

Passaggio 1: specificare le integrazioni di servizi simulati in un file di configurazione fittizio

Puoi testare Step Functions AWS SDK e integrazioni di servizi ottimizzate utilizzando Step Functions Local. L'immagine seguente mostra la macchina a stati definita nella scheda Definizione della macchina a stati:



A tale scopo, è necessario creare un file di configurazione fittizio contenente le sezioni definite in. Introduzione alla struttura della configurazione fittizia

1. Crea un file denominato `MockConfigFile.json` per configurare i test con integrazioni di servizi simulate.

L'esempio seguente mostra un file di configurazione fittizio che fa riferimento a una macchina a stati con due stati definiti denominati and. `LambdaState` `SQSState`

Mock configuration file example

Di seguito è riportato un esempio di file di configurazione fittizio che dimostra come simulare le risposte richiamando una funzione Lambda e inviando un messaggio ad Amazon SQS. In questo esempio, la macchina a `LambdaSQSIntegration`stati contiene tre casi di test denominati`HappyPath`,`RetryPath`, e `HybridPath` che simulano gli stati denominati and. `Task` `LambdaState` `SQSState` Questi stati utilizzano le risposte di `MockedLambdaSuccess` servizio `MockedSQSSuccess` simulate e `MockedLambdaRetry` simulate. Queste risposte di servizio simulate sono definite nella `MockedResponses` sezione del 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

Di seguito è riportato un esempio di definizione di macchina a stati denominata`LambdaSQSIntegration`, che definisce due stati delle attività di integrazione dei servizi denominati `LambdaState` and. `SQSState` `LambdaState`contiene una politica di ripetizione dei tentativi basata su. `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
    }
  }
```

```
}
```

È possibile eseguire la definizione della macchina a `LambdaSQSIntegration` stati a cui si fa riferimento nel file di configurazione fittizio utilizzando uno dei seguenti casi di test:

- `HappyPath`- Questo test simula rispettivamente l'output e l'`SQSState`utilizzo `MockedLambdaSuccess` di `LambdaState` e. `MockedSQSSuccess`
  - `LambdaState`Restituirà il seguente valore:

```
"0":{
  "Return":{
    "StatusCode":200,
    "Payload":{
      "StatusCode":200,
      "body":"Hello from Lambda!"
    }
  }
}
```

  - `SQSState`Restituirà il seguente valore:

```
"0":{
  "Return":{
    "MD5OfMessageBody":"3bcb6e8e-7h85-4375-b0bc-1a59812c6e51",
    "MessageId":"3bcb6e8e-8b51-4375-b0bc-1a59812c6e51"
  }
}
```

- `RetryPath`- Questo test simula `MockedSQSSuccess` rispettivamente l'output `LambdaState` e `SQSState` l'utilizzo di `MockedLambdaRetry` e. Inoltre, `LambdaState` è configurato per eseguire quattro tentativi di nuovo tentativo. Le risposte simulate per questi tentativi sono definite e indicizzate nello stato. `MockedLambdaRetry`
  - Il tentativo iniziale termina con un'operazione non riuscita contenente un messaggio di causa e di errore, come illustrato nell'esempio seguente:

```
"0":{
  "Throw": {
    "Error": "Lambda.ResourceNotReadyException",
    "Cause": "Lambda resource is not ready."
  }
```

```
}
```

- Il primo e il secondo tentativo terminano con un errore dell'operazione contenente un messaggio di causa e di errore, come illustrato nell'esempio seguente:

```
"1-2":{
  "Throw": {
    "Error": "Lambda.TimeoutException",
    "Cause": "Lambda timed out."
  }
}
```

- Il terzo tentativo termina con un'operazione riuscita contenente il risultato dello stato della sezione Payload nella risposta Lambda simulata.

```
"3":{
  "Return": {
    "StatusCode": 200,
    "Payload": {
      "StatusCode": 200,
      "body": "Hello from Lambda!"
    }
  }
}
```

> **Note**
>
> - Per gli stati con una politica di nuovi tentativi, Step Functions Local esaurirà i tentativi di riprova impostati nella politica fino a quando non riceverà una risposta positiva. Ciò significa che, per i tentativi ripetuti, è necessario indicare un numero di tentativi consecutivi e includere tutti i tentativi ripetuti prima di restituire una risposta riuscita.
> - Se non si specifica una risposta simulata per un tentativo specifico, ad esempio, riprova «3", l'esecuzione della macchina a stati avrà esito negativo.

- `HybridPath`- Questo test simula l'output di. `LambdaState` Dopo aver `LambdaState` eseguito correttamente e ricevuto dati simulati come risposta, `SQSState` esegue una chiamata di servizio effettiva alla risorsa specificata in produzione.

Per informazioni su come avviare esecuzioni di test con integrazioni di servizi simulate, consulta. Fase 3: Eseguire i test di integrazione dei servizi simulati

2. Assicurati che la struttura delle risposte simulate sia conforme alla struttura delle risposte di servizio effettive che ricevi quando effettui chiamate di servizio integrate. Per informazioni sui requisiti strutturali per le risposte simulate, vedere. Configurazione di integrazioni di servizi fittizie

   Nell'esempio precedente, il file di configurazione fittizio, le risposte simulate sono definite `MockedLambdaSuccess` e sono `MockedLambdaRetry` conformi alla struttura delle risposte effettive restituite dalla chiamata. `HelloFromLambda`

   > **Important**
   >
   > AWS le risposte al servizio possono variare nella struttura tra i diversi servizi. Step Functions Local non verifica se le strutture di risposta simulate sono conformi alle strutture di risposta del servizio effettive. È necessario assicurarsi che le risposte simulate siano conformi alle risposte effettive prima del test. Per esaminare la struttura delle risposte di servizio, è possibile eseguire le chiamate di servizio effettive utilizzando Step Functions o visualizzare la documentazione relativa a tali servizi.

Passaggio 2: fornire il file di configurazione fittizio a Step Functions Local

È possibile fornire il file di configurazione fittizio a Step Functions Local in uno dei seguenti modi:

Docker

> **Note**
>
> Se utilizzi la versione Docker di Step Functions Local, puoi fornire il file di configurazione fittizio utilizzando solo una variabile di ambiente. Inoltre, è necessario montare il file di configurazione fittizio nel contenitore Step Functions Local all'avvio iniziale del server.

Monta il file di configurazione fittizio su qualsiasi directory all'interno del contenitore Step Functions Local. Quindi, imposta una variabile di ambiente denominata `SFN_MOCK_CONFIG` che contiene il percorso del file di configurazione fittizio nel contenitore. Questo metodo consente di

assegnare qualsiasi nome al file di configurazione fittizio purché la variabile di ambiente contenga il percorso e il nome del file.

Il comando seguente mostra il formato per avviare l'immagine 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
```

L'esempio seguente utilizza il comando per avviare l'immagine 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

Utilizzate uno dei seguenti modi per fornire il file di configurazione fittizio a Step Functions Local:

- Posiziona il file di configurazione fittizio nella stessa directory di. `Step FunctionsLocal.jar` Quando si utilizza questo metodo, è necessario assegnare un nome al file di configurazione fittizio. `MockConfigFile.json`
- Nella sessione che esegue Step Functions Local, imposta una variabile di ambiente denominata`SFN_MOCK_CONFIG`, sul percorso completo del file di configurazione fittizio. Questo metodo consente di denominare il file di configurazione fittizio con qualsiasi nome purché la variabile di ambiente contenga il percorso e il nome del file. Nell'esempio seguente, la `SFN_MOCK_CONFIG` variabile è impostata in modo che punti a un file di configurazione fittizio denominato`EnvSpecifiedMockConfig.json`, che si trova nella `/home/workspace` directory.

  ```
  export SFN_MOCK_CONFIG="/home/workspace/EnvSpecifiedMockConfig.json"
  ```

> **Note**
>
> - Se non si fornisce la variabile di ambiente `SFN_MOCK_CONFIG` a Step Functions Local, per impostazione predefinita, tenterà di leggere un file di configurazione fittizio denominato `MockConfigFile.json` nella directory da cui è stato avviato Step Functions Local.
> - Se si posiziona il file di configurazione fittizio nella stessa directory `Step FunctionsLocal.jar` e si imposta la variabile di ambiente`SFN_MOCK_CONFIG`, Step Functions Local leggerà il file specificato dalla variabile di ambiente.

Fase 3: Eseguire i test di integrazione dei servizi simulati

Dopo aver creato e fornito un file di configurazione fittizio a Step Functions Local, esegui la macchina a stati configurata nel file di configurazione fittizio utilizzando integrazioni di servizi fittizi. Quindi controlla i risultati dell'esecuzione utilizzando un'azione API.

1. Crea una macchina a stati basata sulla definizione menzionata in precedenza nel file di configurazione fittizio.

```
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. Esegui la macchina a stati utilizzando integrazioni di servizi simulate.

   Per utilizzare il file di configurazione fittizio, effettua una chiamata `StartExecution` API su una macchina a stati configurata nel file di configurazione fittizio. A tale scopo, aggiungi

il suffisso,#*test_name*, all'ARN della macchina a stati utilizzata da. `StartExecution` *test_name*è un test case, configurato per la macchina a stati nello stesso file di configurazione fittizio.

Il comando seguente è un esempio che utilizza la macchina a `LambdaSQSIntegration` stati e la configurazione fittizia. In questo esempio, la macchina a `LambdaSQSIntegration` stati viene eseguita utilizzando il `HappyPath` test definito inPassaggio 1: specificare le integrazioni di servizi simulati in un file di configurazione fittizio. Il `HappyPath` test contiene la configurazione per l'esecuzione per gestire le chiamate di integrazione di servizi fittizi effettuate `LambdaState` e `SQSState` dagli stati utilizzando le risposte del `MockedLambdaSuccess` servizio `MockedSQSSuccess` simulate.

```
aws stepfunctions start-execution \
    --endpoint http://localhost:8083 \
    --name executionWithHappyPathMockedServices \
    --state-machine arn:aws:states:us-
east-1:123456789012:stateMachine:LambdaSQSIntegration#HappyPath
```

3. Visualizza la risposta di esecuzione della macchina a stati.

   La risposta alla chiamata `StartExecution` utilizzando un test di integrazione del servizio simulato è la stessa della risposta alla chiamata `StartExecution` normale, che restituisce l'ARN di esecuzione e la data di inizio.

   Di seguito è riportato un esempio di risposta alla chiamata `StartExecution` utilizzando il test di integrazione del servizio simulato:

   ```
   {
      "startDate":"2022-01-28T15:03:16.981000-05:00",
      "executionArn":"arn:aws:states:us-
   east-1:123456789012:execution:LambdaSQSIntegration:executionWithHappyPathMockedServices"
   }
   ```

4. Controlla i risultati dell'esecuzione effettuando una chiamata `ListExecutionsDescribeExecution`, o `GetExecutionHistory` API.

   ```
   aws stepfunctions get-execution-history \
       --endpoint http://localhost:8083 \
       --execution-arn arn:aws:states:us-
   east-1:123456789012:execution:LambdaSQSIntegration:executionWithHappyPathMockedServices
   ```

L'esempio seguente mostra parti di una risposta alla chiamata `GetExecutionHistory` utilizzando l'ARN di esecuzione dalla risposta di esempio mostrata nel passaggio 2. In questo esempio, l'output di `LambdaState` e `SQSState` è costituito dai dati fittizi definiti nel `MockedLambdaSuccess` e `MockedSQSSuccess` nel file di configurazione fittizio. Inoltre, i dati simulati vengono utilizzati nello stesso modo in cui verrebbero utilizzati i dati restituiti eseguendo chiamate di integrazione dei servizi effettive. Inoltre, in questo esempio, l'output da `LambdaState` viene passato `SQSState` come 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 di configurazione per integrazioni di servizi fittizi

Per utilizzare integrazioni di servizi fittizie, devi prima creare un file di configurazione fittizio denominato `MockConfigFile.json` contenente le tue configurazioni fittizie. Quindi fornisci a Step Functions Local il file di configurazione fittizio. Questo file di configurazione definisce i casi di test, che contengono stati fittizi che utilizzano risposte simulate di integrazione dei servizi. La sezione seguente contiene informazioni sulla struttura della configurazione fittizia che include gli stati simulati e le risposte simulate:

Argomenti

- Introduzione alla struttura della configurazione fittizia
- Configurazione di integrazioni di servizi fittizie

Introduzione alla struttura della configurazione fittizia

Una configurazione fittizia è un oggetto JSON contenente i seguenti campi di primo livello:

- `StateMachines`- I campi di questo oggetto rappresentano macchine a stati configurate per utilizzare integrazioni di servizi simulate.
- `MockedResponse`- I campi di questo oggetto rappresentano risposte simulate per le chiamate di integrazione dei servizi.

Di seguito è riportato un esempio di file di configurazione fittizio che include una `StateMachine` definizione e`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!"
        }
      }
    }
  }
}
```

```
}
```

Riferimento al campo di configurazione fittizia

Le sezioni seguenti spiegano i campi oggetto di primo livello che è necessario definire nella configurazione fittizia.

- StateMachines
- MockedResponses

StateMachines

L'`StateMachines`oggetto definisce quali macchine a stati utilizzeranno integrazioni di servizi simulate. La configurazione per ogni macchina a stati è rappresentata come un campo di primo livello di`StateMachines`. Il nome del campo è il nome della macchina a stati e il valore è un oggetto contenente un singolo campo denominato`TestCases`, i cui campi rappresentano i casi di test di quella macchina a stati.

La sintassi seguente mostra una macchina a stati con due casi di test:

```
"MyStateMachine": {
  "TestCases": {
    "HappyPath": {
      ...
    },
    "SadPath": {
      ...
    }
  }
```

TestCases

I campi di `TestCases` rappresentano i singoli casi di test per la macchina a stati. Il nome di ogni test case deve essere univoco per macchina a stati e il valore di ogni test case è un oggetto che specifica una risposta simulata da utilizzare per gli stati Task nella macchina a stati.

L'esempio seguente di a `TestCase` collega due `Task` stati a due`MockedResponses`:

```
"HappyPath": {
  "SomeTaskState": "SomeMockedResponse",
  "AnotherTaskState": "AnotherMockedResponse"
```

```
}
```

MockedResponses

`MockedResponses`è un oggetto contenente più oggetti di risposta simulati con nomi di campo univoci. Un oggetto di risposta simulato definisce il risultato positivo o l'output di errore per ogni chiamata di uno stato Task simulato. Si specifica il numero di chiamata utilizzando singole stringhe intere, come «0», «1», «2» e «3» o un intervallo inclusivo di numeri interi, come «0-1», «2-3».

Quando si simula un Task, è necessario specificare una risposta simulata per ogni chiamata. Una risposta deve contenere un singolo campo denominato `Return` o `Throw` il cui valore è il risultato o l'errore di output della falsa chiamata Task. Se non si specifica una risposta fittizia, l'esecuzione della macchina a stati fallirà.

Di seguito è riportato un esempio di `Return` oggetti `MockedResponse` con `Throw` e. In questo esempio, le prime tre volte che viene eseguita la macchina a stati, `"0-2"` viene restituita la risposta specificata in e la quarta volta che viene eseguita la macchina a stati, `"3"` viene restituita la risposta specificata in.

```
"SomeMockedResponse": {
  "0-2": {
    "Throw": {
      ...
    }
  },
  "3": {
    "Return": {
      ...
    }
  }
}
```

> **Note**
>
> Se utilizzi uno `Map` stato e desideri garantire risposte prevedibili per lo `Map` stato, imposta il valore su 1. `maxConcurrency` Se imposti un valore maggiore di 1, Step Functions Local eseguirà più iterazioni contemporaneamente, il che renderà imprevedibile l'ordine complessivo di esecuzione degli stati tra le iterazioni. Ciò può inoltre far sì che Step Functions Local utilizzi diverse risposte simulate per gli stati di iterazione da un'esecuzione all'altra.

Return

`Return`è rappresentato come un campo degli `MockedResponse` oggetti. Specifica il risultato positivo di uno stato Task simulato.

Di seguito è riportato un esempio di `Return` oggetto che contiene una risposta simulata per la chiamata `Invoke`a una funzione Lambda:

```
"Return": {
  "StatusCode": 200,
  "Payload": {
    "StatusCode": 200,
    "body": "Hello from Lambda!"
  }
}
```

Lancia

`Throw`è rappresentato come un campo degli `MockedResponse` oggetti. Specifica l'output di errore di un'operazione fallita. Il valore di `Throw` deve essere un oggetto contenente `Cause` campi `Error` e con valori di stringa. Inoltre, il valore della stringa specificato nel `Error` campo in `MockConfigFile.json` deve corrispondere agli errori gestiti nelle `Catch` sezioni `Retry` e e della macchina a stati.

Di seguito è riportato un esempio di `Throw` oggetto che contiene una risposta simulata per la chiamata `Invoke`a una funzione Lambda:

```
"Throw": {
  "Error": "Lambda.TimeoutException",
  "Cause": "Lambda timed out."
}
```

Configurazione di integrazioni di servizi fittizie

Puoi simulare qualsiasi integrazione di servizi utilizzando Step Functions Local. Tuttavia, Step Functions Local non impone che i mock siano gli stessi delle API reali. Un Task simulato non chiamerà mai l'endpoint del servizio. Se non si specifica una risposta fittizia, un Task tenterà di chiamare gli endpoint del servizio. Inoltre, Step Functions Local genererà automaticamente un task token quando si simula un task utilizzando il `.waitForTaskToken`.

Le migliori pratiche per Step Functions

Le seguenti best practice per l'implementazione dei flussi di lavoro di AWS Step Functions possono aiutarti a ottimizzare le prestazioni delle applicazioni.

Argomenti

- Utilizza i timeout per evitare esecuzioni bloccate
- Usa gli ARN Amazon S3 invece di trasferire payload di grandi dimensioni
- Evita di raggiungere la quota di cronologia
- Gestire le eccezioni del servizio Lambda
- Evita la latenza durante i sondaggi relativi alle attività
- Scelta di flussi di lavoro Standard o Express
- Limitazioni relative alle dimensioni della politica delle risorse di Amazon CloudWatch Logs

Utilizza i timeout per evitare esecuzioni bloccate

Per impostazione predefinita, Amazon States Language non specifica i timeout per le definizioni delle macchine a stati. Senza un timeout esplicito, Step Functions spesso si affida esclusivamente alla risposta di un addetto all'attività per sapere che un'attività è completa. Se qualcosa va storto e il `TimeoutSeconds` campo non è specificato per uno `Task` stato `Activity` o, l'esecuzione rimane bloccata in attesa di una risposta che non arriverà mai.

Per evitare questa situazione, specifica un timeout ragionevole quando crei una macchina a stati `Task` nella tua macchina a stati. Ad esempio:

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

Se si utilizza un callback con un token task (. waitForTaskToken), ti consigliamo di utilizzare heartbeats e di aggiungere il `HeartbeatSeconds` campo nella definizione dello stato. `Task` Puoi

impostare un valore inferiore `HeartbeatSeconds` al timeout dell'attività, in modo che se il flusso di lavoro fallisce con un errore di battito cardiaco, saprai che è dovuto all'errore dell'attività e non al completamento dell'operazione che richiede molto tempo.

```
{
  "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
    }
  }
}
```

Per ulteriori informazioni, consulta la documentazione Attività di Amazon States Language.

> **Note**
>
> Puoi impostare un timeout per la tua macchina a stati utilizzando il `TimeoutSeconds` campo nella definizione di Amazon States Language. Per ulteriori informazioni, consulta Struttura della macchina a stati.

Usa gli ARN Amazon S3 invece di trasferire payload di grandi dimensioni

Le esecuzioni che trasmettono payload di dati di grandi dimensioni da uno stato all'altro si possono terminare. Se i dati che trasmetti tra gli stati potrebbero superare i 256 KB, usa Amazon Simple Storage Service (Amazon S3) per archiviare i dati e analizza l'Amazon Resource Name (ARN) del bucket nel parametro per ottenere il nome del bucket e `Payload` il valore della chiave. In alternativa modifica la tua applicazione in modo da trasferire nelle esecuzioni payload di dimensioni più contenute.

Nell'esempio seguente, una macchina a stati passa l'input a una AWS Lambda funzione, che elabora un file JSON in un bucket Amazon S3. Dopo aver eseguito questa macchina a stati, la funzione Lambda legge il contenuto del file JSON e lo restituisce come output.

Creazione della funzione Lambda

La seguente funzione Lambda denominata *`pass-large-payload`* legge il contenuto di un file JSON archiviato in uno specifico bucket Amazon S3.

> **Note**
>
> Dopo aver creato questa funzione Lambda, assicurati di fornire al suo ruolo IAM l'autorizzazione appropriata per leggere da un bucket Amazon S3. Ad esempio, collega l'ReadOnlyAccessautorizzazione AmazonS3 al ruolo della funzione 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
```

Crea la macchina a stati

La seguente macchina a stati richiama la funzione Lambda creata in precedenza.

```
{
   "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
      }
   }
}
```

Invece di passare una grande quantità di dati in input, puoi salvarli in un bucket Amazon S3 e passare l'Amazon Resource Name (ARN) del bucket nel `Payload` parametro per ottenere il nome del bucket e il valore della chiave. La funzione Lambda può quindi utilizzare quell'ARN per accedere direttamente ai dati. Di seguito è riportato un esempio di input per l'esecuzione della macchina a stati, in cui i dati vengono archiviati `data.json` in un bucket Amazon S3 denominato. *large-payload-json*

```
{
  "key": "data.json",
  "bucket": "arn:aws:s3:::large-payload-json"
}
```

Evita di raggiungere la quota di cronologia

AWS Step Functionsha una quota fissa di 25.000 voci nella cronologia degli eventi di esecuzione. Quando un'esecuzione raggiunge 24.999 eventi, attende che si verifichi l'evento successivo.

- Se l'evento numero 25.000 è`ExecutionSucceeded`, l'esecuzione viene completata correttamente.
- Se l'evento numero 25.000 non lo è`ExecutionSucceeded`, l'`ExecutionFailed`evento viene registrato e l'esecuzione della macchina a stati fallisce a causa del raggiungimento del limite cronologico

Per evitare di raggiungere questa quota per le esecuzioni di lunga durata, puoi provare una delle seguenti soluzioni alternative:

- Usa lo stato della mappa in modalità Distribuita. In questa modalità, lo `Map` stato esegue ogni iterazione come esecuzione di workflow secondario, il che consente un'elevata concorrenza di un massimo di 10.000 esecuzioni parallele di workflow secondari. Ogni esecuzione del flusso di lavoro secondario ha una propria cronologia di esecuzione separata da quella del flusso di lavoro principale.
- Avvia l'esecuzione di una nuova macchina a stati direttamente dallo `Task` stato di un'esecuzione in esecuzione. Per avviare tali esecuzioni di workflow annidate, utilizzate l'azione `StartExecution` API di Step Functions nella macchina a stati principale insieme ai parametri necessari. Per ulteriori informazioni sull'utilizzo dei flussi di lavoro annidati, consulta Avvio delle esecuzioni di flusso di lavoro da uno stato di task o Utilizzo di un'azione dell'API Step Functions per continuare un nuovo tutorial sull'esecuzione.

  > **Tip**
  >
  > Per distribuire un esempio di flusso di lavoro nidificato sul tuoAccount AWS, consulta Modulo 13 - Flussi di lavoro Nested Express.

- Implementate uno schema che utilizzi una AWS Lambda funzione in grado di avviare una nuova esecuzione della macchina a stati per suddividere il lavoro in corso tra più esecuzioni del flusso di lavoro. Per maggiori informazioni, vedi il tutorial Utilizzo di una funzione Lambda per continuare una nuova esecuzione.

Gestire le eccezioni del servizio Lambda

AWS Lambda presenta occasionalmente errori di servizio transitori. In questo caso, l'invocazione di Lambda genera un errore 500, ad `ClientExecutionTimeoutException` esempio`ServiceException`,`AWSLambdaException`, o. `SdkClientException` Come best

practice, gestisci in modo proattivo queste eccezioni nella tua macchina a stati per `Retry` richiamare la funzione Lambda o l'errore. `Catch`

Gli errori Lambda vengono segnalati come. `Lambda.`*`ErrorName`* Per riprovare un errore di eccezione del servizio Lambda, puoi utilizzare il `Retry` codice seguente.

```
"Retry": [ {
   "ErrorEquals": [ "Lambda.ClientExecutionTimeoutException",
 "Lambda.ServiceException", "Lambda.AWSLambdaException", "Lambda.SdkClientException"],
   "IntervalSeconds": 2,
   "MaxAttempts": 6,
   "BackoffRate": 2
} ]
```

> **Note**
>
> Gli errori non gestiti in Lambda vengono riportati come `Lambda.Unknown` nell'output degli errori. Questi includono out-of-memory errori e timeout delle funzioni. È possibile abbinare o `States.TaskFailed` gestire questi errori. `Lambda.Unknown` `States.ALL` Quando Lambda raggiunge il numero massimo di chiamate, l'errore è. `Lambda.TooManyRequestsException` Per ulteriori informazioni sugli errori della funzione Lambda, consulta Gestione degli errori e tentativi automatici nella Developer Guide. AWS Lambda

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Riprovare dopo un errore
- Gestione delle condizioni di errore utilizzando una macchina a stati Step Functions
- Errori di richiamo Lambda

Evita la latenza durante i sondaggi relativi alle attività

L'API `GetActivityTask` è stata progettata per fornire un `taskToken` un'unica volta. Se un `taskToken` viene rimosso durante la comunicazione con un lavoratore di attività, un numero di richieste di `GetActivityTask` può rimanere bloccato per 60 secondi in attesa di risposta finché non si verifica un timeout del `GetActivityTask`.

Se hai solo un numero ridotto di poll in attesa di risposta, è possibile che tutte le richieste si accodino alla richiesta bloccata e si arrestino. Tuttavia, se hai un gran numero di sondaggi in sospeso per ogni attività Amazon Resource Name (ARN) e una certa percentuale delle tue richieste è bloccata in attesa, ce ne saranno molte altre che possono ancora ottenere `taskToken` un risultato e iniziare a essere elaborate.

Per i sistemi di produzione, consigliamo almeno 100 poll aperti per ARN di attività in ogni momento. Se un poll si blocca e una parte di quei poll vi si accoda, rimangono ancora molte altre richieste che otterranno un `taskToken` per elaborare il lavoro mentre la richiesta di `GetActivityTask` è bloccata.

Per evitare questo tipo di problemi di latenza durante il polling dei task:

- Implementa i tuoi poller come thread separati dal lavoro nella tua implementazione del lavoratore di attività.
- Fai in modo di avere almeno 100 poll aperti per ARN di attività in ogni momento.

> **Note**
>
> Il ridimensionamento a 100 poll aperti per ARN può essere costoso. Ad esempio, il polling di 100 funzioni Lambda per ARN è 100 volte più costoso rispetto a una singola funzione Lambda con 100 thread di polling. Per ridurre la latenza e minimizzare i costi, utilizza un linguaggio che dispone di I/O asincrono e implementa più thread di polling per lavoratore. Per vedere un esempio di lavoratore di attività in cui i thread dei poller sono distinti dai thread del lavoro, consultare Esempio di lavoratore di attività in Ruby.

Per ulteriori informazioni sulle attività e sui lavoratori di attività, consultare Attività.

Scelta di flussi di lavoro Standard o Express

AWS Step Functions offre flussi di lavoro Standard come tipo di flusso di lavoro predefinito, con la possibilità di scegliere flussi di lavoro Express.

Puoi scegliere Flussi di lavoro standard quando hai bisogno di flussi di lavoro di lunga durata, durevoli e verificabili, oppure Express Workflows per carichi di lavoro di elaborazione di eventi ad alto volume. Le esecuzioni della macchina a stati si comportano in modo diverso, a seconda di cosa si seleziona per `Type`. Una volta creata la macchina a stati, non è possibile modificare l'opzione `Type` scelta.

- Per informazioni dettagliate sulle differenze tra i flussi di lavoro Standard ed Express, consulta. Flussi di lavoro Standard e Express
- Per informazioni sull'ottimizzazione dei costi durante la creazione di flussi di lavoro serverless utilizzando Step Functions, vedere. Ottimizzazione dei costi con Express Workflows

Limitazioni relative alle dimensioni della politica delle risorse di Amazon CloudWatch Logs

CloudWatch Le politiche relative alle risorse di Logs sono limitate a 5120 caratteri. Quando CloudWatch Logs rileva che una policy si avvicina a questo limite di dimensione, abilita automaticamente i gruppi di log che iniziano con. `/aws/vendedlogs/`

Quando si crea una macchina a stati con la registrazione abilitata, Step Functions deve aggiornare la politica delle risorse CloudWatch Logs con il gruppo di log specificato. Per evitare di raggiungere il limite di dimensione della politica delle risorse CloudWatch Logs, inserisci come prefisso i nomi dei gruppi di CloudWatch log Logs con. `/aws/vendedlogs/` Quando si crea un gruppo di log nella console Step Functions, i nomi dei gruppi di log hanno il prefisso. `/aws/vendedlogs/states` Per ulteriori informazioni, vedere Abilitazione della registrazione da determinati AWS servizi.

Se non riesci ad accedere ai CloudWatch registri, consulta Unable to access the CloudWatch Logs per informazioni.

Utilizzo AWS Step Functions con altri servizi

Scopri come coordinare altre API di terze parti Servizi AWS o chiamarle con. AWS Step Functions

Argomenti

- Chiama altri servizi AWS
- AWS Integrazioni di servizi SDK
- Integrazioni ottimizzate per Step Functions
- Chiama API di terze parti
- Modelli di integrazione dei servizi
- Passa i parametri a un'API di servizio
- Registro delle modifiche per le integrazioni AWS SDK supportate

Chiama altri servizi AWS

Con le integrazioni di AWS servizi, puoi richiamare azioni API e coordinare le esecuzioni direttamente dal tuo flusso di lavoro. Puoi utilizzare le integrazioni AWS SDK di Step Functions per chiamare uno qualsiasi degli oltre duecento AWS servizi direttamente dalla tua macchina a stati, dandoti accesso a oltre novemila azioni API. Oppure puoi utilizzare le integrazioni ottimizzate di Step Functions, ognuna delle quali è stata personalizzata per fornire funzionalità speciali per il tuo flusso di lavoro. Alcune azioni API sono disponibili in entrambi i tipi di integrazione. Quando possibile, consigliamo di utilizzare l'integrazione ottimizzata.

Puoi coordinare questi servizi direttamente da uno `Task` stato in Amazon States Language. Ad esempio, utilizzando Step Functions, puoi chiamare altri servizi per:

- Invoca una AWS Lambda funzione.
- Esegui un AWS Batch lavoro e quindi esegui azioni diverse in base ai risultati.
- Inserisci o ottieni un articolo da Amazon DynamoDB.
- Esegui un'attività Amazon Elastic Container Service (Amazon ECS) e attendi che venga completata.
- Pubblica su un argomento in Amazon Simple Notification Service (Amazon SNS).
- Invia un messaggio in Amazon Simple Queue Service (Amazon SQS).
- Gestisci un lavoro AWS Glue per Amazon SageMaker.

- Crea flussi di lavoro per l'esecuzione di job Amazon EMR.
- Avvia l'esecuzione di un AWS Step Functions flusso di lavoro.

Integrazioni ottimizzate

Le integrazioni ottimizzate sono state personalizzate da Step Functions per fornire funzionalità speciali per un contesto di flusso di lavoro. Ad esempio, Lambda `Invoke` converte l'output dell'API da un oggetto JSON in escape a un oggetto JSON. AWS Batch`SubmitJob`consente di sospendere l'esecuzione fino al completamento del processo. Il primo set di integrazioni ottimizzate è stato rilasciato nel 2018 e ora ci sono più di cinquanta API.

AWS Integrazioni SDK

AWS Le integrazioni SDK funzionano esattamente come una chiamata API standard che utilizza l'SDK. AWS Offrono la possibilità di richiamare oltre novemila API tra gli oltre duecento AWS servizi direttamente dalla definizione della macchina a stati.

Supporto per modelli di integrazione

I flussi di lavoro standard e i flussi di lavoro Express supportano le stesse integrazioni ma non gli stessi modelli di integrazione.

- Il supporto dei modelli di integrazione ottimizzati è diverso per ogni integrazione.
- I flussi di lavoro Express non supportano Run a Job (.sync) o Wait for Callback (. waitForTaskToken).
- Per ulteriori informazioni, consulta Flussi di lavoro Standard e Express.

Standard Workflows

Integrazioni di servizi supportate

	Servizio	Richiesta e risposta	Esegui un job (`.sync`)	Attendi ilcallback (`.waitForTaskToken`)
Integrazioni	Amazon API Gateway	✓		✓

	Servizio	Richiesta e risposta	Esegui un job (`.sync`)	Attendi il callback (`.waitForTaskToken`)
ottimizzate	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	✓	✓	
	Amazon SageMaker	✓	✓	
	Amazon SNS	✓		✓
	Amazon SQS	✓		✓
	AWS Step Functions	✓	✓	✓

	Servizio	Richiesta e risposta	Esegui un job (**.sync**)	Attendi il callback (**.waitForTaskToken**)
AWS Integrazioni SDK	Oltre duecento	✓		✓

Express Workflows

Integrazioni di servizi supportate

	Servizio	Richiesta e risposta	Esegui un job (**.sync**)	Attendi il callback (**.waitForTaskToken**)
Integrazioni ottimizzate	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	✓		

	Servizio	Richiesta e risposta	Esegui un job (`.sync`)	Attendi il callback (`.waitForTaskToken`)
	Amazon EventBridge	✓		
	AWS Glue	✓		
	AWS Glue DataBrew	✓		
	AWS Lambda	✓		
	AWS Elemental MediaConvert	✓		
	Amazon SageMaker	✓		
	Amazon SNS	✓		
	Amazon SQS	✓		
	AWS Step Functions	✓		
AWS Integrazioni SDK	Oltre duecento	✓		

Accesso multi-account

Step Functions fornisce l'accesso tra account a risorse configurate Account AWS in diversi flussi di lavoro. Utilizzando le integrazioni dei servizi Step Functions, puoi richiamare qualsiasi AWS risorsa tra account anche se questa Servizio AWS non supporta policy basate sulle risorse o chiamate tra account.

Per ulteriori informazioni, consulta Accesso a risorse Account AWS in altri flussi di lavoro.

AWS Integrazioni di servizi SDK

AWS Step Functions si integra con Servizi AWS, consentendoti di richiamare le azioni API di ciascun servizio dal tuo flusso di lavoro. Puoi utilizzare le integrazioni AWS SDK di Step Functions per richiamare quasi tutte Servizio AWS le azioni API dalla tua macchina a stati. Puoi anche utilizzare le integrazioni ottimizzate di Step Functions, ognuna delle quali è stata personalizzata per fornire funzionalità speciali per il tuo flusso di lavoro.

Alcuni servizi o SDK potrebbero non essere disponibili come integrazioni AWS SDK, temporaneamente o permanentemente. I servizi rilasciati di recente potrebbero non avere interazioni SDK disponibili fino a un aggiornamento successivo. Alcuni servizi richiedono una configurazione personalizzata, ad esempio la specifica di un endpoint specifico del cliente, che potrebbe essere più adatto per un'integrazione ottimizzata. Altri SDK non sono adatti all'uso in un flusso di lavoro, come quelli per lo streaming di audio o video. Infine, alcuni servizi potrebbero essere sospesi fino a quando non superano determinate convalide interne eseguite da Step Functions.

> **Tip**
>
> Per implementare un esempio di flusso di lavoro che utilizza integrazioni AWS SDK sul tuo computer Account AWS, consulta il Modulo 9 - Integrazioni dei servizi AWS SDK in The Workshop. AWS Step Functions

Argomenti

- Utilizzo delle integrazioni dei servizi SDK AWS
- Integrazioni di servizi AWS SDK supportate
- Azioni API non supportate per i servizi supportati
- Integrazioni di servizi SDK obsolete AWS

Utilizzo delle integrazioni dei servizi SDK AWS

Per utilizzare le integrazioni AWS SDK, è necessario specificare il nome del servizio e la chiamata API e, facoltativamente, un modello di integrazione del servizio.

> **Note**
>
> - I parametri in Step Functions sono espressi in PascalCase, anche se l'API del servizio nativo è in CamelCase. Ad esempio, è possibile utilizzare l'azione Step Functions API `startSyncExecution` e specificare il relativo parametro come. `StateMachineArn`
> - Per le azioni API che accettano parametri enumerati, come l'azione `DescribeLaunchTemplateVersions` API per Amazon EC2, specifica una versione plurale del nome del parametro. Ad esempio, specifica il `Filter.N` parametro `Filters` dell'azione API. `DescribeLaunchTemplateVersions`

Puoi chiamare i servizi AWS SDK direttamente da Amazon States Language nel `Resource` campo dello stato di un'attività. Per fare ciò, usa la seguente sintassi:

`arn:aws:states:::aws-sdk:`*`serviceName`*`:`*`apiAction`*`.`*`[serviceIntegrationPattern]`*

Ad esempio, per Amazon EC2, potresti usare. `arn:aws:states:::aws-sdk:ec2:describeInstances` Ciò restituirebbe l'output definito per la chiamata API DescribeInstances di Amazon EC2.

Se un'integrazione AWS SDK rileva un errore, il campo Error risultante sarà composto dal nome del servizio e dal nome dell'errore, separati da un punto:. *`ServiceName`*`.`*`ErrorName`* Sia il nome del servizio che il nome dell'errore sono in formato Pascal. È inoltre possibile visualizzare il nome del servizio, in lettere minuscole, nel campo Risorsa dello stato dell'attività. Puoi trovare i nomi dei potenziali errori nella documentazione di riferimento dell'API del servizio di destinazione.

Ad esempio, potresti utilizzare l'integrazione `arn:aws:states:::aws-sdk:acmpca:deleteCertificateAuthority` AWS SDK. L'AWS Private Certificate Authority API Reference indica che l'azione `DeleteCertificateAuthority` API può generare, ad esempio`ResourceNotFoundException`, un. Per gestire questo errore, dovreste quindi specificare l'errore `AcmPca.ResourceNotFoundException` nei Retrier o Catcher dello stato dell'operazione. Per ulteriori informazioni sulla gestione degli errori in Step Functions, vedereGestione degli errori in Step Functions.

Step Functions non può generare automaticamente le policy IAM per AWS le integrazioni SDK. Dopo aver creato la macchina a stati, dovrai accedere alla console IAM e configurare le policy relative ai ruoli. Per ulteriori informazioni, consulta Politiche IAM per servizi integrati.

Consulta il Raccogli informazioni sui bucket Amazon S3 utilizzando AWS le integrazioni dei servizi SDK tutorial per un esempio di come utilizzare le integrazioni AWS SDK.

Integrazioni di servizi AWS SDK supportate

La tabella seguente elenca le integrazioni dei servizi AWS SDK supportate da Step Functions. La colonna *Task*State Resource elenca la sintassi per richiamare un'azione API specifica quando si utilizza l'integrazione per il servizio specificato nella colonna Service name a sinistra. La colonna Data supportata fornisce informazioni sulle date in cui è stata supportata l'integrazione del servizio. Inoltre, la colonna del prefisso Exception sulla destra elenca i prefissi di eccezione per ogni integrazione di servizio. Questi prefissi di eccezione sono presenti nelle eccezioni generate quando si esegue erroneamente un'integrazione del AWS servizio SDK con Step Functions.

Note

- I servizi contrassegnati con *** hanno azioni API che al momento non sono supportate da Step Functions. Per informazioni sulle azioni che non sono supportate per un servizio, consulta la Azioni API non supportate per i servizi supportati tabella.
- Per informazioni sugli aggiornamenti apportati a ogni lancio per espandere il supporto per le integrazioni AWS SDK, consulta. Registro delle modifiche per le integrazioni AWS SDK supportate

Important

Il supporto all'azione delle API viene rilasciato con cadenza trimestrale. Gli aggiornamenti alle azioni già supportate, come i nuovi parametri, potrebbero non essere immediatamente disponibili.

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
AWS AppFabric	`arn:aws:states:::aws-sdk:appfabric: [apiAction]`	18 gennaio 2024	AppFabric

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
B2B Data Interchange	`arn:aws:states:::aws-sdk:b2bi: [apiAction]`	18 gennaio 2024	B2Bi
Esportazioni di dati AWS	`arn:aws:states:::aws-sdk:bcmdataexports: [apiAction]`	18 gennaio 2024	BcmDataExports
Amazon Bedrock	`arn:aws:states:::aws-sdk:bedrock: [apiAction]`	18 gennaio 2024	Bedrock
Amazon Bedrock Agents	`arn:aws:states:::aws-sdk:bedrockagent: [apiAction]`	18 gennaio 2024	BedrockAgent
Amazon Bedrock Runtime Agents	`arn:aws:states:::aws-sdk:bedrockagentruntime: [apiAction]`	18 gennaio 2024	BedrockAgentRuntime
Amazon Bedrock Runtime	`arn:aws:states:::aws-sdk:bedrockruntime: [apiAction]`	18 gennaio 2024	BedrockRuntime
AWS Clean Rooms	`arn:aws:states:::aws-sdk:cleanroomsml: [apiAction]`	18 gennaio 2024	CleanRoomsMl
Amazon CloudFront KeyValueStore	`arn:aws:states:::aws-sdk:cloudfrontkeyvaluestore: [apiAction]`	18 gennaio 2024	CloudFrontKeyValueStore

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
CodeGuru Security	`arn:aws:states:::aws-sdk:codegurusecurity: [apiAction]`	18 gennaio 2024	CodeGuruSecurity
Centrale ottimizzazione costi AWS	`arn:aws:states:::aws-sdk:costoptimizationhub: [apiAction]`	18 gennaio 2024	CostOptimizationHub
Amazon DataZone	`arn:aws:states:::aws-sdk:datazone: [apiAction]`	18 gennaio 2024	DataZone
Amazon EKS Auth	`arn:aws:states:::aws-sdk:eksauth: [apiAction]`	18 gennaio 2024	EksAuth
AWS Entity Resolution	`arn:aws:states:::aws-sdk:entityresolution: [apiAction]`	18 gennaio 2024	EntityResolution
Piano gratuito di AWS	`arn:aws:states:::aws-sdk:freetier: [apiAction]`	18 gennaio 2024	FreeTier
Amazon Inspector Scan	`arn:aws:states:::aws-sdk:inspectorscan: [apiAction]`	18 gennaio 2024	InspectorScan
AWS Launch Wizard	`arn:aws:states:::aws-sdk:launchwizard: [apiAction]`	18 gennaio 2024	LaunchWizard

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
Amazon Managed Blockchain Query	`arn:aws:states:::aws-sdk:managedblockchainquery: [apiAction]`	18 gennaio 2024	ManagedBlockchainQuery
AWS Elemental MediaPackage V2	`arn:aws:states:::aws-sdk:mediapackagev2: [apiAction]`	18 gennaio 2024	MediaPackageV2
AWS HealthImaging	`arn:aws:states:::aws-sdk:medicalimaging: [apiAction]`	18 gennaio 2024	MedicalImaging
Network Manager	`arn:aws:states:::aws-sdk:networkmanager: [apiAction]`	18 gennaio 2024	NetworkManager
AWS Payment Cryptography	`arn:aws:states:::aws-sdk:paymentcryptography: [apiAction]`	18 gennaio 2024	PaymentCryptography
AWS Payment Cryptography Data	`arn:aws:states:::aws-sdk:paymentcryptographydata: [apiAction]`	18 gennaio 2024	PaymentCryptographyData
AWS Private CA Connector for Active Directory	`arn:aws:states:::aws-sdk:pcaconnectorad: [apiAction]`	18 gennaio 2024	PcaConnectorAd
Amazon Q Business	`arn:aws:states:::aws-sdk:qbusiness: [apiAction]`	18 gennaio 2024	QBusiness

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
Amazon Q Connect	`arn:aws:states:::aws-sdk:qconnect: [apiAction]`	18 gennaio 2024	QConnect
AWS re:Post	`arn:aws:states:::aws-sdk:repostspace: [apiAction]`	18 gennaio 2024	Repostspace
Amazon Timestream Query	`arn:aws:states:::aws-sdk:timestreamquery: [apiAction]`	18 gennaio 2024	TimestreamQuery
Amazon Timestream Write	`arn:aws:states:::aws-sdk:timestreamwrite: [apiAction]`	18 gennaio 2024	TimestreamWrite
Trusted Advisor	`arn:aws:states:::aws-sdk:trustedadvisor: [apiAction]`	18 gennaio 2024	TrustedAdvisor
Verified Permissions	`arn:aws:states:::aws-sdk:verifiedpermissions: [apiAction]`	18 gennaio 2024	VerifiedPermissions
Amazon WorkSpaces Thin Client	`arn:aws:states:::aws-sdk:workspacesthinclient: [apiAction]`	18 gennaio 2024	WorkSpacesThinClient
AWS CloudTrail Data	`arn:aws:states:::aws-sdk:cloudtraildata: [apiAction]`	16 giugno 2023	CloudTrailData

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
Amazon CloudWatch Internet Monitor	`arn:aws:states:::aws-sdk:internetmonitor: [apiAction]`	16 giugno 2023	InternetMonitor
Amazon Interactive Video Service RealTime	`arn:aws:states:::aws-sdk:ivsrealtime: [apiAction]`	16 giugno 2023	IvsRealTime
AWS IoT TwinMaker	`arn:aws:states:::aws-sdk:iottwinmaker: [apiAction]`	16 giugno 2023	IoTTwinMaker
Amazon OpenSearch Ingestion	`arn:aws:states:::aws-sdk:osis: [apiAction]`	16 giugno 2023	Osis
AWS Telco Network Builder	`arn:aws:states:::aws-sdk: tnb:[apiAction]`	16 giugno 2023	Tnb
Amazon VPC Lattice	`arn:aws:states:::aws-sdk:vpclattice: [apiAction]`	16 giugno 2023	VpcLattice
Amazon Chime Media Pipelines	`arn:aws:states:::aws-sdk:chimesdkmediapipelines: [apiAction]`	17 febbraio 2023	ChimeSdkMediaPipelines
Amazon Chime Voice	`arn:aws:states:::aws-sdk:chimesdkvoice: [apiAction]`	17 febbraio 2023	ChimeSdkVoice

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
Amazon CodeCatalyst	`arn:aws:states:::aws-sdk:codecatalyst:` *[apiAction]*	17 febbraio 2023	CodeCatalyst
Amazon Connect Cases	`arn:aws:states:::aws-sdk:connectcases:` *[apiAction]*	17 febbraio 2023	ConnectCases
Amazon DocumentDB Elastic Clusters	`arn:aws:states:::aws-sdk:docdbelastic:` *[apiAction]*	17 febbraio 2023	DocDbElastic
Amazon EMR Serverless	`arn:aws:states:::aws-sdk:emrserverless:` *[apiAction]*	17 febbraio 2023	EmrServerless
Amazon IVS Chat	`arn:aws:states:::aws-sdk:ivs:` *[apiAction]*	17 febbraio 2023	Ivs
Amazon Kendra Intelligent Ranking	`arn:aws:states:::aws-sdk:kendraranking:` *[apiAction]*	17 febbraio 2023	KendraRanking
AWS HealthOmics	`arn:aws:states:::aws-sdk:omics:` *[apiAction]*	17 febbraio 2023	Omics
Amazon Redshift Serverless	`arn:aws:states:::aws-sdk:redshiftserverless:` *[apiAction]*	17 febbraio 2023	RedshiftServerless
Amazon Security Lake	`arn:aws:states:::aws-sdk:securitylake:` *[apiAction]*	17 febbraio 2023	SecurityLake

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
AWS Backup Storage	`arn:aws:states:::aws-sdk:backupstorage: [apiAction]`	17 febbraio 2023	BackupStorage
AWS Clean Rooms	`arn:aws:states:::aws-sdk:cleanrooms: [apiAction]`	17 febbraio 2023	CleanRooms
AWS Control Tower	`arn:aws:states:::aws-sdk:controltower: [apiAction]`	17 febbraio 2023	ControlTower
AWS Health	`arn:aws:states:::aws-sdk:health: [apiAction]`	17 febbraio 2023	Health
AWS IoT FleetWise	`arn:aws:states:::aws-sdk:iotfleetwise: [apiAction]`	17 febbraio 2023	IotFleetWise
AWS IoT RoboRunner	`arn:aws:states:::aws-sdk:iotroborunner: [apiAction]`	17 febbraio 2023	IotRoboRunner
AWS Mainframe Modernization	`arn:aws:states:::aws-sdk:m2: [apiAction]`	17 febbraio 2023	M2
Orchestratore dell'Hub di migrazione AWS	`arn:aws:states:::aws-sdk:migrationhuborchestrator: [apiAction]`	17 febbraio 2023	Migration HubOrchestrator

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
AWS Private 5G	`arn:aws:states:::aws-sdk:privatenetworks: [apiAction]`	17 febbraio 2023	PrivateNetworks
Esploratore di risorse AWS	`arn:aws:states:::aws-sdk:resourceexplorer2: [apiAction]`	17 febbraio 2023	ResourceExplorer2
AWS SimSpace Weaver	`arn:aws:states:::aws-sdk:simspaceweaver: [apiAction]`	17 febbraio 2023	SimSpaceWeaver
AWS Support App	`arn:aws:states:::aws-sdk:supportapp: [apiAction]`	17 febbraio 2023	SupportApp
CloudWatch Observability Access Manager	`arn:aws:states:::aws-sdk:oam: [apiAction]`	17 febbraio 2023	Oam
EventBridge Pipes	`arn:aws:states:::aws-sdk:pipes: [apiAction]`	17 febbraio 2023	Pipes
EventBridge Scheduler	`arn:aws:states:::aws-sdk:scheduler: [apiAction]`	17 febbraio 2023	Scheduler
IAM Roles Anywhere	`arn:aws:states:::aws-sdk:rolesanywhere: [apiAction]`	17 febbraio 2023	RolesAnywhere

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
Kinesis Video WebRTC Storage	`arn:aws:states:::aws-sdk:kinesisvideowebrtcstorage: [apiAction]`	17 febbraio 2023	KinesisVideoWebRtcStorage
License Manager Linux Subscriptions	`arn:aws:states:::aws-sdk:licensemanagerlinuxsubscriptions: [apiAction]`	17 febbraio 2023	LicenseManagerLinuxSubscriptions
License Manager User Subscriptions	`arn:aws:states:::aws-sdk:licensemanagerusersubscriptions: [apiAction]`	17 febbraio 2023	LicenseManagerUserSubscriptions
OpenSearch Serverless	`arn:aws:states:::aws-sdk:opensearchserverless: [apiAction]`	17 febbraio 2023	OpenSearchServerless
Route 53 ARC Zonal Shift	`arn:aws:states:::aws-sdk:arczonalshift: [apiAction]`	17 febbraio 2023	ArcZonalShift
SageMaker Geospatial	`arn:aws:states:::aws-sdk:sagemakergeospatial: [apiAction]`	17 febbraio 2023	SageMaker Geospatial
SageMaker Metrics	`arn:aws:states:::aws-sdk:sagemakermetrics: [apiAction]`	17 febbraio 2023	SageMakerMetrics

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
Systems Manager for SAP	`arn:aws:states:::aws-sdk:ssmsap:` *[apiAction]*	17 febbraio 2023	SsmSap
AWS Account Management	`arn:aws:states:::aws-sdk:account:` *[apiAction]*	19 aprile 2022	Account
AWS Amplify	`arn:aws:states:::aws-sdk:amplify:` *[apiAction]*	30 settembre 2021	Amplify
AWS App Mesh	`arn:aws:states:::aws-sdk:appmesh:` *[apiAction]*	30 settembre 2021	AppMesh
AWS App Runner	`arn:aws:states:::aws-sdk:apprunner:` *[apiAction]*	30 settembre 2021	AppRunner
AWS AppConfig	`arn:aws:states:::aws-sdk:appconfig:` *[apiAction]*	30 settembre 2021	AppConfig
AWS AppConfig Data	`arn:aws:states:::aws-sdk:appconfigdata:` *[apiAction]*	19 aprile 2022	AppConfigData
AWS AppSync	`arn:aws:states:::aws-sdk:appsync:` *[apiAction]*	30 settembre 2021	AppSync

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
AWS Application Discovery Service	`arn:aws:states:::aws-sdk:applicationdiscovery:` *`[apiAction]`* ***	30 settembre 2021	ApplicationDiscovery
AWS Application Migration Service	`arn:aws:states:::aws-sdk:mgn:` *`[apiAction]`*	30 settembre 2021	Mgn
AWS Audit Manager	`arn:aws:states:::aws-sdk:auditmanager:` *`[apiAction]`*	30 settembre 2021	AuditManager
AWS Auto Scaling Plans	`arn:aws:states:::aws-sdk:autoscalingplans:` *`[apiAction]`*	30 settembre 2021	AutoScalingPlans
AWS Backup	`arn:aws:states:::aws-sdk:backup:` *`[apiAction]`*	30 settembre 2021	Backup
AWS Backup gateway	`arn:aws:states:::aws-sdk:backupgateway:` *`[apiAction]`*	19 aprile 2022	BackupGateway
AWS Batch	`arn:aws:states:::aws-sdk:batch:` *`[apiAction]`*	30 settembre 2021	Batch
AWS Billing Conductor	`arn:aws:states:::aws-sdk:billingconductor:` *`[apiAction]`*	26 luglio 2022	Billingconductor

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
Budget AWS	`arn:aws:states:::aws-sdk:budgets: [apiAction]`	30 settembre 2021	Budgets
AWS Certificate Manager	`arn:aws:states:::aws-sdk:acm: [apiAction]`	30 settembre 2021	Acm
AWS Private Certificate Authority	`arn:aws:states:::aws-sdk:acmpca: [apiAction]`	30 settembre 2021	AcmPca
AWS Cloud Map	`arn:aws:states:::aws-sdk:servicediscovery: [apiAction]`	30 settembre 2021	ServiceDiscovery
AWS Cloud9	`arn:aws:states:::aws-sdk:cloud9: [apiAction]`	30 settembre 2021	Cloud9
AWS CloudFormation	`arn:aws:states:::aws-sdk:cloudformation: [apiAction]`	30 settembre 2021	CloudFormation
AWS CloudHSM	`arn:aws:states:::aws-sdk:cloudhsm: [apiAction]`	30 settembre 2021	CloudHsm
AWS CloudHSM	`arn:aws:states:::aws-sdk:cloudhsmv2: [apiAction]`	30 settembre 2021	CloudHsmV2
AWS CloudTrail	`arn:aws:states:::aws-sdk:cloudtrail: [apiAction]`	30 settembre 2021	CloudTrail

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
AWS Cloud Control	`arn:aws:states:::aws-sdk:cloudcontrol: [apiAction]`	19 aprile 2022	CloudControl
AWS CodeBuild	`arn:aws:states:::aws-sdk:codebuild: [apiAction]`	30 settembre 2021	CodeBuild
AWS CodeCommit	`arn:aws:states:::aws-sdk:codecommit: [apiAction]`	30 settembre 2021	CodeCommit
AWS CodeDeploy	`arn:aws:states:::aws-sdk:codedeploy: [apiAction]` ***	30 settembre 2021	CodeDeploy
AWS CodePipeline	`arn:aws:states:::aws-sdk:codepipeline: [apiAction]`	30 settembre 2021	CodePipeline
AWS CodeStar	`arn:aws:states:::aws-sdk:codestar: [apiAction]`	30 settembre 2021	CodeStar
AWS CodeStar	`arn:aws:states:::aws-sdk:codestarnotifications: [apiAction]`	30 settembre 2021	CodestarNotifications
AWS CodeStar	`arn:aws:states:::aws-sdk:codestarconnections: [apiAction]`	30 settembre 2021	CodeStarConnections

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
AWS Compute Optimizer	`arn:aws:states:::aws-sdk:computeoptimizer: [apiAction]`	30 settembre 2021	ComputeOptimizer
AWS Config	`arn:aws:states:::aws-sdk:config: [apiAction]`	30 settembre 2021	Config
AWS Cost Explorer Service	`arn:aws:states:::aws-sdk:costexplorer: [apiAction]`	30 settembre 2021	CostExplorer
AWS Cost and Usage Report	`arn:aws:states:::aws-sdk:costandusagereport: [apiAction]`	30 settembre 2021	CostAndUsageReport
AWS Data Exchange	`arn:aws:states:::aws-sdk:dataexchange: [apiAction]`	30 settembre 2021	DataExchange
AWS Data Pipeline	`arn:aws:states:::aws-sdk:datapipeline: [apiAction]`	30 settembre 2021	DataPipeline
AWS DataSync	`arn:aws:states:::aws-sdk:datasync: [apiAction]`	30 settembre 2021	DataSync
AWS Database Migration Service	`arn:aws:states:::aws-sdk:databasemigration: [apiAction]`	30 settembre 2021	DatabaseMigration
AWS Device Farm	`arn:aws:states:::aws-sdk:devicefarm: [apiAction]`	30 settembre 2021	DeviceFarm

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
AWS Direct Connect	`arn:aws:states:::aws-sdk:directconnect: [apiAction]` *** —	30 settembre 2021	DirectConnect
AWS Directory Service	`arn:aws:states:::aws-sdk:directory: [apiAction]`	30 settembre 2021	Directory
AWS EC2 Instance Connect	`arn:aws:states:::aws-sdk:ec2instanceconnect: [apiAction]`	30 settembre 2021	Ec2InstanceConnect
AWS Elastic Beanstalk	`arn:aws:states:::aws-sdk:elasticbeanstalk: [apiAction]`	30 settembre 2021	ElasticBeanstalk
AWS Elemental MediaLive	`arn:aws:states:::aws-sdk:medialive: [apiAction]`	30 settembre 2021	MediaLive
AWS Elemental MediaPackage	`arn:aws:states:::aws-sdk:mediapackage: [apiAction]` *** —	30 settembre 2021	MediaPackage
AWS Elemental MediaPackage VOD	`arn:aws:states:::aws-sdk:mediapackagevod: [apiAction]`	30 settembre 2021	MediaPackageVod
AWS Elemental MediaStore	`arn:aws:states:::aws-sdk:mediastore: [apiAction]`	30 settembre 2021	MediaStore
AWS Fault Injection Service	`arn:aws:states:::aws-sdk:fis: [apiAction]`	30 settembre 2021	Fis

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
AWS Firewall Manager	`arn:aws:states:::aws-sdk:fms:` *`[apiAction]`*	30 settembre 2021	Fms
AWS Glue	`arn:aws:states:::aws-sdk:glue:` *`[apiAction]`*	30 settembre 2021	Glue
AWS Glue DataBrew	`arn:aws:states:::aws-sdk:databrew:` *`[apiAction]`*	30 settembre 2021	DataBrew
AWS IoT Greengrass	`arn:aws:states:::aws-sdk:greengrass:` *`[apiAction]`*	30 settembre 2021	Greengrass
AWS Ground Station	`arn:aws:states:::aws-sdk:groundstation:` *`[apiAction]`*	30 settembre 2021	GroundStation
AWS Identity and Access Management	`arn:aws:states:::aws-sdk:iam:` *`[apiAction]`*	30 settembre 2021	Iam
AWS IoT	`arn:aws:states:::aws-sdk:iot:` *`[apiAction]`* ***	30 settembre 2021	Iot
AWS IoT 1-Click	`arn:aws:states:::aws-sdk:iot1clickprojects:` *`[apiAction]`*	30 settembre 2021	Iot1ClickProjects
AWS IoT Analytics	`arn:aws:states:::aws-sdk:iotanalytics:` *`[apiAction]`*	30 settembre 2021	IoTAnalytics

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
AWS IoT Core Device Advisor	`arn:aws:states:::aws-sdk:iotdeviceadvisor: [apiAction]` *** —	30 settembre 2021	IotDeviceAdvisor
AWS IoT Events	`arn:aws:states:::aws-sdk:iotevents: [apiAction]`	30 settembre 2021	IotEvents
Dati Eventi AWS IoT	`arn:aws:states:::aws-sdk:ioteventsdata: [apiAction]`	30 settembre 2021	IotEventsData
AWS IoT Fleet Hub	`arn:aws:states:::aws-sdk:iotfleethub: [apiAction]`	30 settembre 2021	IoTFleetHub
AWS IoT Greengrass Version 2	`arn:aws:states:::aws-sdk:greengrassv2: [apiAction]`	30 settembre 2021	GreengrassV2
Dati processi AWS IoT Plane	`arn:aws:states:::aws-sdk:iotjobsdataplane: [apiAction]`	30 settembre 2021	IotJobsDataPlane
AWS IoT Secure Tunneling	`arn:aws:states:::aws-sdk:iotsecuretunneling: [apiAction]`	30 settembre 2021	IoTSecure Tunneling
AWS IoT SiteWise	`arn:aws:states:::aws-sdk:iotsitewise: [apiAction]`	30 settembre 2021	IoTSiteWise
AWS IoT Wireless	`arn:aws:states:::aws-sdk:iotwireless: [apiAction]`	30 settembre 2021	IotWireless

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
AWS Key Management Service	`arn:aws:states:::aws-sdk:kms:` *`[apiAction]`*	30 settembre 2021	Kms
AWS Lake Formation	`arn:aws:states:::aws-sdk:lakeformation:` *`[apiAction]`*	30 settembre 2021	LakeFormation
AWS Lambda	`arn:aws:states:::aws-sdk:lambda:` *`[apiAction]`* ***	30 settembre 2021	Lambda
AWS License Manager	`arn:aws:states:::aws-sdk:licensemanager:` *`[apiAction]`*	30 settembre 2021	LicenseManager
Marketplace AWS	`arn:aws:states:::aws-sdk:marketplacecatalog:` *`[apiAction]`*	30 settembre 2021	MarketplaceCatalog
AWS Marketplace Commerce Analytics	`arn:aws:states:::aws-sdk:marketplacecommerceanalytics:` *`[apiAction]`*	30 settembre 2021	MarketplaceCommerceAnalytics
AWS Marketplace Entitlement Service	`arn:aws:states:::aws-sdk:marketplaceentitlement:` *`[apiAction]`*	30 settembre 2021	MarketplaceEntitlement
AWS Elemental MediaTailor	`arn:aws:states:::aws-sdk:mediatailor:` *`[apiAction]`*	30 settembre 2021	MediaTailor

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
AWS Migration Hub	`arn:aws:states:::aws-sdk:migrationhub: [apiAction]`	30 settembre 2021	MigrationHub
AWS Migration Hub Config	`arn:aws:states:::aws-sdk:migrationhubconfig: [apiAction]`	30 settembre 2021	MigrationHubConfig
Suggerimenti di strategia dell'Hub di migrazione AWS	`arn:aws:states:::aws-sdk:migrationhubstrategy: [apiAction]`	19 aprile 2022	MigrationHubStrategy
AWS Mobile	`arn:aws:states:::aws-sdk:mobile: [apiAction]`	30 settembre 2021	
AWS Network Firewall	`arn:aws:states:::aws-sdk:networkfirewall: [apiAction]`	30 settembre 2021	NetworkFirewall
AWS OpsWorks	`arn:aws:states:::aws-sdk:opsworks: [apiAction]`	30 settembre 2021	OpsWorks
AWS OpsWorks CM	`arn:aws:states:::aws-sdk:opsworkscm: [apiAction]`	30 settembre 2021	OpsWorksCm
AWS Organizations	`arn:aws:states:::aws-sdk:organizations: [apiAction]`	30 settembre 2021	Organizations

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
AWS Outposts	`arn:aws:states:::aws-sdk:outposts: [apiAction]`	30 settembre 2021	Outposts
AWS Panorama	`arn:aws:states:::aws-sdk:panorama: [apiAction]`	19 aprile 2022	Panorama
Amazon Relational Database Service Performance Insights	`arn:aws:states:::aws-sdk:pi: [apiAction]`	30 settembre 2021	Pi
Listino prezzi AWS	`arn:aws:states:::aws-sdk:pricing: [apiAction]`	30 settembre 2021	Pricing
Amazon Relational Database Service	`arn:aws:states:::aws-sdk:rdsdata: [apiAction]` ***	30 settembre 2021	RdsData
AWS Resilience Hub	`arn:aws:states:::aws-sdk:resiliencehub: [apiAction]`	19 aprile 2022	Resiliencehub
AWS Resource Access Manager	`arn:aws:states:::aws-sdk:ram: [apiAction]`	30 settembre 2021	Ram
AWS Resource Groups	`arn:aws:states:::aws-sdk:resourcegroups: [apiAction]`	30 settembre 2021	ResourceGroups

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
AWS Resource Groups Tagging API	`arn:aws:states:::aws-sdk:resourcegroupstaggingapi: [apiAction]`	30 settembre 2021	ResourceGroupsTaggingApi
AWS RoboMaker	`arn:aws:states:::aws-sdk:robomaker: [apiAction]`	30 settembre 2021	RoboMaker
AWS IAM Identity Center	`arn:aws:states:::aws-sdk:identitystore: [apiAction]`	30 settembre 2021	Identitystore
IAM Identity Center OIDC	`arn:aws:states:::aws-sdk:ssooidc: [apiAction]`	30 settembre 2021	SsoOidc
AWS Secrets Manager	`arn:aws:states:::aws-sdk:secretsmanager: [apiAction]`	30 settembre 2021	SecretsManager
AWS Security Token Service	`arn:aws:states:::aws-sdk:sts: [apiAction]` *** —	30 settembre 2021	Sts
AWS Security Hub	`arn:aws:states:::aws-sdk:securityhub: [apiAction]`	30 settembre 2021	SecurityHub
AWS Server Migration Service	`arn:aws:states:::aws-sdk:sms: [apiAction]`	30 settembre 2021	Sms

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
AWS Service Catalog	`arn:aws:states:::aws-sdk:servicecatalog: [apiAction]`	30 settembre 2021	ServiceCatalog
AWS Service Catalog AppRegistry	`arn:aws:states:::aws-sdk:servicecatalogappregistry: [apiAction]`	30 settembre 2021	ServiceCatalogAppRegistry
AWS Shield	`arn:aws:states:::aws-sdk:shield: [apiAction]` *** —	30 settembre 2021	Shield
AWS Signer	`arn:aws:states:::aws-sdk:signer: [apiAction]`	30 settembre 2021	Signer
IAM Identity Center	`arn:aws:states:::aws-sdk:sso: [apiAction]`	30 settembre 2021	Sso
IAM Identity Center Admin	`arn:aws:states:::aws-sdk:ssoadmin: [apiAction]`	30 settembre 2021	SsoAdmin
AWS Step Functions	`arn:aws:states:::aws-sdk:sfn: [apiAction]`	30 settembre 2021	Sfn
AWS Storage Gateway	`arn:aws:states:::aws-sdk:storagegateway: [apiAction]`	30 settembre 2021	StorageGateway

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
AWS Support	`arn:aws:states:::aws-sdk:support: [apiAction]`	30 settembre 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 settembre 2021	Transfer
AWS WAF	`arn:aws:states:::aws-sdk:waf: [apiAction]`	30 settembre 2021	Waf
AWS WAF Regional	`arn:aws:states:::aws-sdk:wafregional: [apiAction]`	30 settembre 2021	WafRegional
AWS WAFV2	`arn:aws:states:::aws-sdk:wafv2: [apiAction]`	30 settembre 2021	Wafv2
AWS Well-Architected Tool	`arn:aws:states:::aws-sdk:wellarchitected: [apiAction]`	30 settembre 2021	WellArchitected
AWS X-Ray	`arn:aws:states:::aws-sdk:xray: [apiAction]`	30 settembre 2021	XRay

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
AWS Marketplace Metering Service	`arn:aws:states:::aws-sdk:marketplacemetering:` *`[apiAction]`*	30 settembre 2021	MarketplaceMetering
AWS Serverless Application Repository	`arn:aws:states:::aws-sdk:serverlessapplicationrepository:` *`[apiAction]`*	30 settembre 2021	ServerlessApplicationRepository
AWS Identity and Access Management Access Analyzer	`arn:aws:states:::aws-sdk:accessanalyzer:` *`[apiAction]`*	30 settembre 2021	AccessAnalyzer
Alexa for Business	`arn:aws:states:::aws-sdk:alexaforbusiness:` *`[apiAction]`*	30 settembre 2021	AlexaForBusiness
Amazon API Gateway	`arn:aws:states:::aws-sdk:apigateway:` *`[apiAction]`*	30 settembre 2021	ApiGateway
Amazon API Gateway	`arn:aws:states:::aws-sdk:apigatewayv2:` *`[apiAction]`*	30 settembre 2021	ApiGatewayV2
Amazon AppIntegrations	`arn:aws:states:::aws-sdk:appintegrations:` *`[apiAction]`*	30 settembre 2021	AppIntegrations
Amazon AppStream 2.0	`arn:aws:states:::aws-sdk:appstream:` *`[apiAction]`*	30 settembre 2021	AppStream

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
Amazon AppFlow	`arn:aws:states:::aws-sdk:appflow: [apiAction]`	30 settembre 2021	Appflow
Amazon Athena	`arn:aws:states:::aws-sdk:athena: [apiAction]`	30 settembre 2021	Athena
Amazon Augmented AI	`arn:aws:states:::aws-sdk:sagemakera2iruntime: [apiAction]`	30 settembre 2021	SageMaker A2IRuntime
Amazon Braket	`arn:aws:states:::aws-sdk:braket: [apiAction]`	30 settembre 2021	Braket
Amazon Chime	`arn:aws:states:::aws-sdk:chime: [apiAction]`	30 settembre 2021	Chime
Amazon Chime Meetings	`arn:aws:states:::aws-sdk:chimesdkmeetings: [apiAction]`	19 aprile 2022	ChimeSdkMeetings
Amazon Cloud Directory	`arn:aws:states:::aws-sdk:clouddirectory: [apiAction]`	30 settembre 2021	CloudDirectory
Amazon CloudFront	`arn:aws:states:::aws-sdk:cloudfront: [apiAction]`	30 settembre 2021	CloudFront

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
Amazon CloudSearch	`arn:aws:states:::aws-sdk:cloudsearch: [apiAction]`	30 settembre 2021	CloudSearch
Amazon CloudWatch	`arn:aws:states:::aws-sdk:cloudwatch: [apiAction]`	30 settembre 2021	CloudWatch
Amazon CloudWatch Application Insights	`arn:aws:states:::aws-sdk:applicationinsights: [apiAction]`	30 settembre 2021	ApplicationInsights
CloudWatch Evidently	`arn:aws:states:::aws-sdk:evidently: [apiAction]`	19 aprile 2022	Evidently
Amazon CloudWatch Logs	`arn:aws:states:::aws-sdk:cloudwatchlogs: [apiAction]`	30 settembre 2021	CloudWatchLogs
Amazon CloudWatch RUM	`arn:aws:states:::aws-sdk:rum: [apiAction]`	19 aprile 2022	Rum
Amazon CloudWatch Synthetics	`arn:aws:states:::aws-sdk:synthetics: [apiAction]`	30 settembre 2021	Synthetics
Amazon CodeGuru Profiler	`arn:aws:states:::aws-sdk:codeguruprofiler: [apiAction]`	30 settembre 2021	CodeGuruProfiler

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
Amazon CodeGuru Reviewer	`arn:aws:states:::aws-sdk:codegurureviewer: [apiAction]`	30 settembre 2021	CodeGuruReviewer
Amazon Cognito	`arn:aws:states:::aws-sdk:cognitoidentity: [apiAction]`	30 settembre 2021	CognitoIdentity
Amazon Cognito Identity Provider	`arn:aws:states:::aws-sdk:cognitoidentityprovider: [apiAction]`	30 settembre 2021	CognitoIdentityProvider
Amazon Cognito Sync	`arn:aws:states:::aws-sdk:cognitosync: [apiAction]`	30 settembre 2021	CognitoSync
Amazon Comprehend	`arn:aws:states:::aws-sdk:comprehend: [apiAction]`	30 settembre 2021	Comprehend
Amazon Comprehend Medical	`arn:aws:states:::aws-sdk:comprehendmedical: [apiAction]` *** —	30 settembre 2021	ComprehendMedical
Amazon Connect Contact Lens	`arn:aws:states:::aws-sdk:connectcontactlens: [apiAction]`	30 settembre 2021	ConnectContactLens
Amazon Connect Participant Service	`arn:aws:states:::aws-sdk:connectparticipant: [apiAction]`	30 settembre 2021	ConnectParticipant

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
Amazon Connect	`arn:aws:states:::aws-sdk:connect: [apiAction]`	30 settembre 2021	Connect
Amazon Connect Voice ID	`arn:aws:states:::aws-sdk:voiceid: [apiAction]`	19 aprile 2022	VoiceId
Amazon Connect Wisdom	`arn:aws:states:::aws-sdk:wisdom: [apiAction]`	19 aprile 2022	Wisdom
Amazon Data Lifecycle Manager	`arn:aws:states:::aws-sdk:dlm: [apiAction]`	30 settembre 2021	Dlm
Amazon Detective	`arn:aws:states:::aws-sdk:detective: [apiAction]`	30 settembre 2021	Detective
Amazon DevOps Guru	`arn:aws:states:::aws-sdk:devopsguru: [apiAction]`	30 settembre 2021	DevOpsGuru
Amazon DocumentDB (with MongoDB compatibility)	`arn:aws:states:::aws-sdk:docdb: [apiAction]`	30 settembre 2021	DocDb
Amazon DynamoDB	`arn:aws:states:::aws-sdk:dynamodb: [apiAction]`	30 settembre 2021	DynamoDb

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
Amazon DynamoDB Streams	`arn:aws:states:::aws-sdk:dynamodbstreams: [apiAction]`	30 settembre 2021	DynamoDbStreams
Amazon EC2 Container Registry	`arn:aws:states:::aws-sdk:ecr: [apiAction]`	30 settembre 2021	Ecr
Amazon EC2 Container Service	`arn:aws:states:::aws-sdk:ecs: [apiAction]`	30 settembre 2021	Ecs
Amazon EC2 Systems Manager	`arn:aws:states:::aws-sdk:ssm: [apiAction]`	30 settembre 2021	Ssm
Amazon EMR	`arn:aws:states:::aws-sdk:emrcontainers: [apiAction]` *** —	30 settembre 2021	EmrContainers
Amazon ElastiCache	`arn:aws:states:::aws-sdk:elasticache: [apiAction]`	30 settembre 2021	ElastiCache
Amazon Elastic Inference	`arn:aws:states:::aws-sdk:elasticinference: [apiAction]`	30 settembre 2021	ElasticInference
Amazon Elastic Block Store	`arn:aws:states:::aws-sdk:ebs: [apiAction]`	30 settembre 2021	Ebs
Amazon Elastic Compute Cloud	`arn:aws:states:::aws-sdk:ec2: [apiAction]`	30 settembre 2021	Ec2

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
Amazon Elastic Container Registry Public	`arn:aws:states:::aws-sdk:ecrpublic: [apiAction]`	30 settembre 2021	EcrPublic
Amazon Elastic File System	`arn:aws:states:::aws-sdk:efs: [apiAction]` ***	30 settembre 2021	Efs
Amazon Elastic Kubernetes Service	`arn:aws:states:::aws-sdk:eks: [apiAction]`	30 settembre 2021	Eks
Amazon EMR	`arn:aws:states:::aws-sdk:emr: [apiAction]`	30 settembre 2021	Emr
Amazon Elastic Transcoder	`arn:aws:states:::aws-sdk:elastictranscoder: [apiAction]` ***	30 settembre 2021	ElasticTranscoder
Amazon OpenSearch Service	`arn:aws:states:::aws-sdk:elasticsearch: [apiAction]`	30 settembre 2021	Elasticsearch
Amazon OpenSearch Service	`arn:aws:states:::aws-sdk:opensearch: [apiAction]`	19 aprile 2022	OpenSearch
Amazon EventBridge	`arn:aws:states:::aws-sdk:eventbridge: [apiAction]`	30 settembre 2021	EventBridge
Amazon FSx	`arn:aws:states:::aws-sdk:fsx: [apiAction]`	30 settembre 2021	FSx

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
Amazon Forecast Query	`arn:aws:states:::aws-sdk:forecastquery: [apiAction]`	30 settembre 2021	Forecastquery
Amazon Forecast Service	`arn:aws:states:::aws-sdk:forecast: [apiAction]`	30 settembre 2021	Forecast
Amazon Fraud Detector	`arn:aws:states:::aws-sdk:frauddetector: [apiAction]`	30 settembre 2021	FraudDetector
Amazon GameLift	`arn:aws:states:::aws-sdk:gamelift: [apiAction]`	30 settembre 2021	Amazon GameLift
Amazon GameSparks	`arn:aws:states:::aws-sdk:gamesparks: [apiAction]`	27 luglio 2022	GameSparks
Amazon S3 Glacier	`arn:aws:states:::aws-sdk:glacier: [apiAction]`	30 settembre 2021	Glacier
Amazon GuardDuty	`arn:aws:states:::aws-sdk:guardduty: [apiAction]`	30 settembre 2021	GuardDuty
AWS HealthLake	`arn:aws:states:::aws-sdk:healthlake: [apiAction]`	30 settembre 2021	HealthLake
Amazon Honeycode	`arn:aws:states:::aws-sdk:honeycode: [apiAction]`	30 settembre 2021	Honeycode

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
Amazon Inspector	`arn:aws:states:::aws-sdk:inspector: [apiAction]`	30 settembre 2021	Inspector
Amazon Inspector V2	`arn:aws:states:::aws-sdk:inspector2: [apiAction]`	19 aprile 2022	Inspector2
Amazon Interactive Video Service	`arn:aws:states:::aws-sdk:ivs: [apiAction]`	30 settembre 2021	Ivs
Amazon Kendra	`arn:aws:states:::aws-sdk:kendra: [apiAction]`	30 settembre 2021	Kendra
Amazon Kinesis	`arn:aws:states:::aws-sdk:kinesis: [apiAction]` *** —	30 settembre 2021	Kinesis
Amazon Kinesis Analytics	`arn:aws:states:::aws-sdk:kinesisanalytics: [apiAction]`	30 settembre 2021	KinesisAnalytics
Amazon Kinesis Analytics V2	`arn:aws:states:::aws-sdk:kinesisanalyticsv2: [apiAction]`	30 settembre 2021	KinesisAnalyticsV2
Amazon Kinesis Firehose	`arn:aws:states:::aws-sdk:firehose: [apiAction]`	30 settembre 2021	Firehose

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
Amazon Kinesis Video Signaling Channels	`arn:aws:states:::aws-sdk:kinesisvideosignaling: [apiAction]`	30 settembre 2021	KinesisVideoSignaling
Amazon Kinesis Video Streams	`arn:aws:states:::aws-sdk:kinesisvideo: [apiAction]`	30 settembre 2021	KinesisVideo
Amazon Kinesis Video Streams Archived Media	`arn:aws:states:::aws-sdk:kinesisvideoarchivedmedia: [apiAction]`	30 settembre 2021	KinesisVideoArchivedMedia
Amazon Kinesis video stream	`arn:aws:states:::aws-sdk:kinesisvideomedia: [apiAction]`	30 settembre 2021	KinesisVideoMedia
Amazon Lex Model Building Service	`arn:aws:states:::aws-sdk:lexmodelbuilding: [apiAction]`	30 settembre 2021	LexModelBuilding
Amazon Lex Model Building Service V2	`arn:aws:states:::aws-sdk:lexmodelsv2: [apiAction]`	30 settembre 2021	LexModelsV2
Amazon Lex	`arn:aws:states:::aws-sdk:lexruntime: [apiAction]`	30 settembre 2021	LexRuntime
Amazon Lex Runtime V2	`arn:aws:states:::aws-sdk:lexruntimev2: [apiAction]` *** —	30 settembre 2021	LexRuntimeV2

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
Amazon Lightsail	`arn:aws:states:::aws-sdk:lightsail:` *[apiAction]*	30 settembre 2021	Lightsail
Servizio di posizione Amazon	`arn:aws:states:::aws-sdk:location:` *[apiAction]*	30 settembre 2021	Location
Amazon Lookout for Equipment	`arn:aws::states:::aws-sdk:lookoutequipment:` *[apiAction]*	30 settembre 2021	LookoutEquipment
Amazon Lookout for Metrics	`arn:aws:states:::aws-sdk:lookoutmetrics:` *[apiAction]*	30 settembre 2021	LookoutMetrics
Amazon Lookout for Vision	`arn:aws:states:::aws-sdk:lookoutvision:` *[apiAction]*	30 settembre 2021	LookoutVision
Amazon MQ	`arn:aws:states:::aws-sdk:mq:` *[apiAction]*	30 settembre 2021	Mq
Amazon Macie	`arn:aws:states:::aws-sdk:macie:` *[apiAction]*	30 settembre 2021	
Amazon Macie 2	`arn:aws:states:::aws-sdk:macie2:` *[apiAction]*	30 settembre 2021	Macie2
Amazon Managed Blockchain	`arn:aws:states:::aws-sdk:managedblockchain:` *[apiAction]*	30 settembre 2021	ManagedBlockchain

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
Amazon Managed Grafana	`arn:aws:states:::aws-sdk:grafana: [apiAction]`	19 aprile 2022	Grafana
Amazon Managed Service for Prometheus	`arn:aws:states:::aws-sdk:amp: [apiAction]`	30 settembre 2021	Amp
Amazon Managed Streaming for Apache Kafka	`arn:aws:states:::aws-sdk:kafka: [apiAction]`	30 settembre 2021	Kafka
Amazon MSK Connect	`arn:aws:states:::aws-sdk:kafkaconnect: [apiAction]`	19 aprile 2022	KafkaConnect
Amazon Managed Workflows for Apache Airflow	`arn:aws:states:::aws-sdk:mwaa: [apiAction]`	30 settembre 2021	Mwaa
Amazon Mechanical Turk	`arn:aws:states:::aws-sdk:mturk: [apiAction]`	30 settembre 2021	MTurk
Amazon MemoryDB for Redis	`arn:aws:states:::aws-sdk:memorydb: [apiAction]`	19 aprile 2022	MemoryDB
Amazon Nimble Studio	`arn:aws:states:::aws-sdk:nimble: [apiAction]`	30 settembre 2021	Nimble
Amazon Personalize	`arn:aws:states:::aws-sdk:personalize: [apiAction]`	30 settembre 2021	Personalize

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
Amazon Personalize Events	`arn:aws:states:::aws-sdk:personalizeevents: [apiAction]`	30 settembre 2021	PersonalizeEvents
Amazon Personalize Runtime	`arn:aws:states:::aws-sdk:personalizeruntime: [apiAction]`	30 settembre 2021	PersonalizeRuntime
Amazon Pinpoint	`arn:aws:states:::aws-sdk:pinpoint: [apiAction]`	30 settembre 2021	Pinpoint
Amazon Pinpoint Email Service	`arn:aws:states:::aws-sdk:pinpointemail: [apiAction]`	30 settembre 2021	PinpointEmail
Amazon Pinpoint SMS and Voice Service	`arn:aws:states:::aws-sdk:pinpointsmsvoice: [apiAction]`	30 settembre 2021	PinpointSmsVoice
Amazon Pinpoint SMS and Voice V2 Service	`arn:aws:states:::aws-sdk:pinpointsmsvoicev2: [apiAction]`	27 luglio 2022	PinpointSmsVoiceV2
Amazon Polly	`arn:aws:states:::aws-sdk:polly: [apiAction]`	30 settembre 2021	Polly
Amazon QLDB	`arn:aws:states:::aws-sdk:qldb: [apiAction]`	30 settembre 2021	Qldb
Amazon QLDB Session	`arn:aws:states:::aws-sdk:qldbsession: [apiAction]`	30 settembre 2021	QldbSession

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
Amazon QuickSight	`arn:aws:states:::aws-sdk:quicksight:` *[apiAction]*	30 settembre 2021	QuickSight
Amazon Redshift	`arn:aws:states:::aws-sdk:redshift:` *[apiAction]*	30 settembre 2021	Redshift
Amazon Redshift Data API	`arn:aws:states:::aws-sdk:redshiftdata:` *[apiAction]*	30 settembre 2021	RedshiftData
Amazon Rekognition	`arn:aws:states:::aws-sdk:rekognition:` *[apiAction]*	30 settembre 2021	Rekognition
Amazon Relational Database Service	`arn:aws:states:::aws-sdk:rds:` *[apiAction]*	30 settembre 2021	Rds
Amazon Route 53	`arn:aws:states:::aws-sdk:route53:` *[apiAction]*	30 settembre 2021	Route53
Amazon Route 53 Recovery Control Config	`arn:aws:states:::aws-sdk:route53recoverycontrolconfig:` *[apiAction]*	30 settembre 2021	Route53RecoveryControlConfig
Amazon Route 53 Domains	`arn:aws:states:::aws-sdk:route53domains:` *[apiAction]*	30 settembre 2021	Route53Domains

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
Amazon Route 53 Resolver	`arn:aws:states:::aws-sdk:route53resolver: [apiAction]`	30 settembre 2021	Route53Resolver
Amazon S3 on Outposts	`arn:aws:states:::aws-sdk:s3outposts: [apiAction]`	30 settembre 2021	S3Outposts
Amazon SageMaker Runtime Feature Store Runtime	`arn:aws:states:::aws-sdk:sagemakerfeaturestoreruntime: [apiAction]`	30 settembre 2021	SageMaker RuntimeFeatureStoreRuntime
Amazon SageMaker Runtime Runtime	`arn:aws:states:::aws-sdk:sagemakerruntime: [apiAction]`	30 settembre 2021	SageMaker RuntimeRuntime
Amazon SageMaker	`arn:aws:states:::aws-sdk:sagemaker: [apiAction]`	30 settembre 2021	SageMaker Runtime
Amazon SageMaker Edge Manager	`arn:aws:states:::aws-sdk:sagemakeredge: [apiAction]`	30 settembre 2021	SagemakerEdge
Amazon Simple Email Service	`arn:aws:states:::aws-sdk:ses: [apiAction]`	30 settembre 2021	Ses
Amazon Simple Email Service V2	`arn:aws:states:::aws-sdk:sesv2: [apiAction]`	30 settembre 2021	SesV2

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
Amazon Simple Notification Service	`arn:aws:states:::aws-sdk:sns:` *`[apiAction]`*	30 settembre 2021	Sns
Amazon Simple Queue Service	`arn:aws:states:::aws-sdk:sqs:` *`[apiAction]`*	30 settembre 2021	Sqs
Amazon Simple Storage Service	`arn:aws:states:::aws-sdk:s3:` *`[apiAction]`* ***	30 settembre 2021	S3
Amazon Simple Workflow Service	`arn:aws:states:::aws-sdk:swf:` *`[apiAction]`*	30 settembre 2021	Swf
Amazon Textract	`arn:aws:states:::aws-sdk:textract:` *`[apiAction]`*	30 settembre 2021	Textract
Amazon Transcribe	`arn:aws:states:::aws-sdk:transcribe:` *`[apiAction]`*	30 settembre 2021	Transcribe
Amazon Translate	`arn:aws:states:::aws-sdk:translate:` *`[apiAction]`*	30 settembre 2021	Translate
Amazon WorkDocs	`arn:aws:states:::aws-sdk:workdocs:` *`[apiAction]`*	30 settembre 2021	WorkDocs
Amazon WorkMail	`arn:aws:states:::aws-sdk:workmail:` *`[apiAction]`*	30 settembre 2021	WorkMail

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
Amazon WorkMail Message Flow	`arn:aws:states:::aws-sdk:workmailmessageflow:` *`[apiAction]`*	30 settembre 2021	WorkMailMessageFlow
Amazon WorkSpaces	`arn:aws:states:::aws-sdk:workspaces:` *`[apiAction]`*	30 settembre 2021	WorkSpaces
Amazon WorkSpaces Web	`arn:aws:states:::aws-sdk:workspacesweb:` *`[apiAction]`*	19 aprile 2022	WorkSpacesWeb
Amplify	`arn:aws:states:::aws-sdk:amplifybackend:` *`[apiAction]`*	30 settembre 2021	AmplifyBackend
Amplify UI Builder	`arn:aws:states:::aws-sdk:amplifyuibuilder:` *`[apiAction]`*	19 aprile 2022	AmplifyUiBuilder
Application Auto Scaling	`arn:aws:states:::aws-sdk:applicationautoscaling:` *`[apiAction]`*	30 settembre 2021	ApplicationAutoScaling
Amazon EC2 Auto Scaling	`arn:aws:states:::aws-sdk:autoscaling:` *`[apiAction]`*	30 settembre 2021	Auto Scaling
CodeArtifact	`arn:aws:states:::aws-sdk:codeartifact:` *`[apiAction]`*	30 settembre 2021	Codeartifact

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
DynamoDB Accelerator	`arn:aws:states:::aws-sdk:dax:` *`[apiAction]`*	30 settembre 2021	Dax
EC2 Image Builder	`arn:aws:states:::aws-sdk:imagebuilder:` *`[apiAction]`*	30 settembre 2021	Imagebuilder
AWS Elastic Disaster Recovery	`arn:aws:states:::aws-sdk:drs:` *`[apiAction]`*	19 aprile 2022	Drs
Elastic Load Balancing	`arn:aws:states:::aws-sdk:elasticloadbalancing:` *`[apiAction]`*	30 settembre 2021	ElasticLoadBalancing
Elastic Load Balancing V2	`arn:aws:states:::aws-sdk:elasticloadbalancingv2:` *`[apiAction]`*	30 settembre 2021	ElasticLoadBalancingV2
MediaConnect	`arn:aws:states:::aws-sdk:mediaconnect:` *`[apiAction]`*	30 settembre 2021	MediaConnect
Amazon S3 Control	`arn:aws:states:::aws-sdk:s3control:` *`[apiAction]`* ***	30 settembre 2021	S3Control
Recycle Bin for Amazon EBS	`arn:aws:states:::aws-sdk:rbin:` *`[apiAction]`*	19 aprile 2022	Rbin

Nome servizio	**Task**risorsa statale	Data supportata	Prefisso di eccezione
Savings Plans	`arn:aws:states:::aws-sdk:savingsplans: [apiAction]`	30 settembre 2021	Savingsplans
Amazon EventBridge Schema Registry	`arn:aws:states:::aws-sdk:schemas: [apiAction]`	30 settembre 2021	Schemas
Service Quotas	`arn:aws:states:::aws-sdk:servicequotas: [apiAction]`	30 settembre 2021	ServiceQuotas
AWS Snowball	`arn:aws:states:::aws-sdk:snowball: [apiAction]`	30 settembre 2021	Snowball

Azioni API non supportate per i servizi supportati

La tabella seguente elenca le azioni API non supportate per le integrazioni dei servizi AWS SDK. La colonna di destra contiene le azioni API che attualmente non sono supportate per il servizio elencato nella colonna di sinistra.

Nome servizio	Azione API non supportata
AWS Application Discovery Service	• `DescribeExportConfigurations` • `ExportConfigurations`
Amazon Bedrock	• `InvokeModelWithResponseStream`
Agenti per Amazon Bedrock Runtime	• `InvokeAgent`
AWS CodeDeploy	• `BatchGetDeploymentInstances` • `GetDeploymentInstance` • `ListDeploymentInstances`

Nome servizio	Azione API non supportata
	• 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 Consulente principale per i dispositivi	• ListTestCases
Amazon Kinesis	• SubscribeToShard
AWS Lambda	• InvokeAsync • InvokeWithResponseStream
Amazon Lex Runtime versione 2	• StartConversation
AWS Elemental MediaPackage	• RotateChannelCredentials
Amazon Relational Database Service	• ExecuteSql

Nome servizio	Azione API non supportata
Amazon Simple Storage Service	• `SelectObjectContent`
Controllo Amazon S3	• `SelectObjectContent`
AWS Shield	• `DeleteSubscription`
AWS Security Token Service	• `AssumeRole` • `AssumeRoleWithSAML` • `AssumeRoleWithWebIdentity`

Integrazioni di servizi SDK obsolete AWS

Le seguenti integrazioni di servizi AWS SDK sono ora obsolete:

- AWS Mobile
- Amazon Macie

Integrazioni ottimizzate per Step Functions

I seguenti argomenti includono le API, i parametri e la sintassi di richiesta/risposta supportati in Amazon States Language per il coordinamento di altri servizi. AWS Viene fornito anche codice di esempio. Puoi chiamare i servizi di integrazione ottimizzata direttamente da Amazon States Language nel `Resource` campo di uno `Task` stato.

È possibile utilizzare tre modelli di integrazione dei servizi:

- Richiedi una risposta (impostazione predefinita): attendi la risposta HTTP, quindi passa allo stato successivo
- Run a Job (`.sync`) - attendi il completamento del lavoro
- Wait for Callback (`.waitForTaskToken`): sospende un flusso di lavoro fino alla restituzione di un task token

I flussi di lavoro standard e i flussi di lavoro Express supportano le stesse integrazioni ma non gli stessi modelli di integrazione.

- Il supporto dei modelli di integrazione ottimizzati è diverso per ogni integrazione.
- I flussi di lavoro Express non supportano Run a Job (.sync) o Wait for Callback (.waitForTaskToken).
- Per ulteriori informazioni, consulta Flussi di lavoro Standard e Express.

Standard Workflows

Integrazioni di servizi supportate

	Servizio	Richiesta e risposta	Esegui un job (`.sync`)	Attendi il callback (`.waitForTaskToken`)
Integrazioni ottimizzate	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	✓	✓	

	Servizio	Richiesta e risposta	Esegui un job (`.sync`)	Attendi il callback (`.waitForTaskToken`)
	AWS Lambda	✓		✓
	AWS Elemental MediaConvert	✓	✓	
	Amazon SageMaker	✓	✓	
	Amazon SNS	✓		✓
	Amazon SQS	✓		✓
	AWS Step Functions	✓	✓	✓
AWS Integrazioni SDK	Oltre duecento	✓		✓

Express Workflows

Integrazioni di servizi supportate

	Servizio	Richiesta e risposta	Esegui un job (`.sync`)	Attendi il callback (`.waitForTaskToken`)
Integrazioni ottimizzate	Amazon API Gateway	✓		
	Amazon Athena	✓		
	AWS Batch	✓		
	Amazon Bedrock	✓		
	AWS CodeBuild	✓		

	Servizio	Richiesta e risposta	Esegui un job (`.sync`)	Attendi il callback (`.waitForTaskToken`)
	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 Integrazioni SDK	Oltre duecento	✓		

Chiama API Gateway con Step Functions

Step Functions può controllare determinati AWS servizi direttamente da Amazon States Language (ASL). Per ulteriori informazioni, consultare Uso di altri servizi e Passa i parametri a un'API di servizio.

> In che modo l'integrazione Optimized API Gateway è diversa dall'integrazione con API Gateway AWS SDK
>
> - `apigateway:invoke:`non ha equivalenti nell'integrazione del servizio AWS SDK. Invece, il servizio Optimized API Gateway chiama direttamente l'endpoint API Gateway.

Utilizzi Amazon API Gateway per creare, pubblicare, gestire e monitorare le API HTTP e REST. Per l'integrazione con API Gateway, definisci uno `Task` stato in Step Functions che richiama direttamente un endpoint API Gateway HTTP o API Gateway REST, senza scrivere codice o fare affidamento su altre infrastrutture. Una definizione di `Task` stato include tutte le informazioni necessarie per la chiamata API. È inoltre possibile selezionare diversi metodi di autorizzazione.

> Note
>
> Step Functions supporta la possibilità di chiamare gli endpoint HTTP tramite API Gateway, ma attualmente non supporta la possibilità di chiamare endpoint HTTP generici.

Supporto della funzionalità API Gateway

L'integrazione di Step Functions API Gateway supporta alcune funzionalità di API Gateway, ma non tutte. Per un elenco più dettagliato delle funzionalità supportate, consulta quanto segue.

- Supportato dalle integrazioni API REST di Step Functions API Gateway e API Gateway HTTP:
  - Autorizzatori: IAM (utilizzando Signature Version 4), No Auth, Lambda Authorizers (basati su parametri di richiesta e basati su token con intestazione personalizzata)
  - Tipi di API: regionali
  - Gestione delle API: nomi di dominio API Gateway API, fase dell'API, percorso, parametri di interrogazione, corpo della richiesta

- Supportato dall'integrazione dell'API HTTP Step Functions API Gateway. L'integrazione dell'API REST di Step Functions API Gateway che offre l'opzione per API ottimizzate per Edge non è supportata.
- Non supportato dall'integrazione con Step Functions API Gateway:
  - Autorizzatori: Amazon Cognito, Native Open ID Connect/OAuth 2.0, intestazione di autorizzazione per autorizzatori Lambda basati su token
  - Tipi di API: private
  - Gestione delle API: nomi di dominio personalizzati

Per ulteriori informazioni su API Gateway e le relative API HTTP e REST, consulta quanto segue.

- La pagina dei concetti di Amazon API Gateway.
- Scelta tra API HTTP e API REST nella guida per sviluppatori di API Gateway.

Formato della richiesta

Quando create la definizione `Task` dello stato, Step Functions convalida i parametri, crea l'URL necessario per eseguire la chiamata, quindi chiama l'API. La risposta include il codice di stato HTTP, le intestazioni e il corpo della risposta. Il formato della richiesta ha parametri obbligatori e facoltativi.

Parametri di richiesta obbligatori

- `ApiEndpoint`
  - Tipo: `String`
  - Il nome host di un URL API Gateway. Il formato è *`<API ID>`*`.execute-api.`*`<region>`*`.amazonaws.com`.

    L'ID API può contenere solo una combinazione dei seguenti caratteri alfanumerici: `0123456789abcdefghijklmnopqrstuvwxyz`
- `Method`
  - Tipo: `Enum`
  - Il metodo HTTP, che deve essere uno dei seguenti:
    - `GET`
    - `POST`
    - `PUT`

- `DELETE`
- `PATCH`
- `HEAD`
- `OPTIONS`

Parametri di richiesta opzionali

- `Headers`
  - Tipo: `JSON`
  - Le intestazioni HTTP consentono un elenco di valori associati alla stessa chiave.
- `Stage`
  - Tipo: `String`
  - Il nome della fase in cui viene distribuita l'API in API Gateway. È facoltativo per qualsiasi API HTTP che utilizza lo `$default` stage.
- `Path`
  - Tipo: `String`
  - Parametri di percorso che vengono aggiunti dopo l'endpoint dell'API.
- `QueryParameters`
  - Tipo: `JSON`
  - Le stringhe di query consentono solo un elenco di valori associati alla stessa chiave.
- `RequestBody`
  - Tipo: `JSON` or `String`
  - Il corpo della richiesta HTTP. Il suo tipo può essere un `JSON` oggetto o`String`. `RequestBody`è supportato solo per `PATCH` i metodi `PUT` HTTP`POST`, e.
- `AllowNullValues`
  - Tipo: `BOOLEAN`
  - L'impostazione `AllowNullValues` su ti `true` consentirà di passare valori nulli come i seguenti:

```
{
    "NewPet": {
        "type": "turtle",
        "price": 123,
        "name": null
```

```
    }
}
```

- `AuthType`
  - Tipo: `JSON`
  - Il metodo di autenticazione. Il metodo predefinito è`NO_AUTH`. I valori consentiti sono:
    - `NO_AUTH`
    - `IAM_ROLE`
    - `RESOURCE_POLICY`

    Per ulteriori informazioni, vedere Autenticazione e autorizzazione.

> **Note**
>
> Per motivi di sicurezza, le seguenti chiavi di intestazione HTTP non sono attualmente consentite:
>
> - Qualsiasi cosa preceduta da, o. `X-Forwarded` `X-Amz` `X-Amzn`
> - `Authorization`
> - `Connection`
> - `Content-md5`
> - `Expect`
> - `Host`
> - `Max-Forwards`
> - `Proxy-Authenticate`
> - `Server`
> - `TE`
> - `Transfer-Encoding`
> - `Trailer`
> - `Upgrade`
> - `Via`
> - `Www-Authenticate`

Il seguente esempio di codice mostra come richiamare API Gateway utilizzando 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"
    }
}
```

Autenticazione e autorizzazione

È possibile utilizzare i seguenti metodi di autenticazione:

- Nessuna autorizzazione: chiama l'API direttamente senza alcun metodo di autorizzazione.
- Ruolo IAM: con questo metodo, Step Functions assume il ruolo di macchina a stati, firma la richiesta con Signature Version 4 (SigV4), quindi chiama l'API.
- Politica delle risorse: Step Functions autentica la richiesta e quindi chiama l'API. È necessario allegare una politica delle risorse all'API che specifichi quanto segue:

  1. La macchina a stati che invocherà API Gateway.

> **Important**
>
> È necessario specificare la macchina a stati per limitarne l'accesso. In caso contrario, verrà concesso l'accesso a qualsiasi macchina a stati che autentica la sua richiesta API Gateway con l'autenticazione delle policy delle risorse sulla vostra API.

2. That Step Functions è il servizio che chiama API Gateway:`"Service": "states.amazonaws.com"`.
3. La risorsa a cui desideri accedere, tra cui:
   - La *regione*.
   - L'*ID dell'account* nella regione specificata.
   - *L'api-id.*
   - *Il nome d'arte.*
   - Il metodo *HTTP-VERB* ().
   - *resource-path-specifier*Il.

Per un esempio di politica delle risorse, consulta le politiche IAM per Step Functions e API Gateway.

Per ulteriori informazioni sul formato delle risorse, consulta Formato delle risorse delle autorizzazioni per l'esecuzione dell'API in API Gateway nella Guida per sviluppatori di API Gateway.

> **Note**
>
> Le policy relative alle risorse sono supportate solo per l'API REST.

Modelli di integrazione dei servizi

L'integrazione con API Gateway supporta due modelli di integrazione dei servizi:

- Richiesta e risposta, che è il modello di integrazione predefinito. Consente a Step Functions di passare alla fase successiva immediatamente dopo aver ricevuto una risposta HTTP.
- Attendere un callback con il token dell'attività(`.waitForTaskToken`), che attende la restituzione di un task token con un payload. Per usare il `.waitForTaskToken` pattern, aggiungi. waitForTaskSimbolo alla fine del campo Risorsa della definizione dell'attività, come mostrato nell'esempio seguente:

```
{
    "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"
    }
}
```

Formato di output

Vengono forniti i seguenti parametri di output:

Nome	Type	Descrizione
`ResponseBody`	`JSON` o `String`	Il corpo della risposta della chiamata API.
`Headers`	`JSON`	Le intestazioni di risposta.
`StatusCode`	`Integer`	Il codice di stato HTTP per la risposta.
`StatusText`	`String`	Il testo dello stato della risposta.

Un esempio di risposta:

```
{
    "ResponseBody": {
        "myBills": []
    },
    "Headers": {
        "key": ["value1", "value2"]
```

```
    },
    "StatusCode": 200,
    "StatusText": "OK"
}
```

Gestione degli errori

Quando si verifica un errore, `cause` viene restituito un `error` and come segue:

- Se il codice di stato HTTP è disponibile, l'errore verrà restituito nel formato`ApiGateway.`*`<HTTP Status Code>`*.
- Se il codice di stato HTTP non è disponibile, l'errore verrà restituito nel formato`ApiGateway.`*`<Exception>`*.

In entrambi i casi, `cause` viene restituito come stringa.

L'esempio seguente mostra una risposta in cui si è verificato un errore:

```
{
    "error": "ApiGateway.403",
    "cause": "{\"message\":\"Missing Authentication Token\"}"
}
```

> **Note**
>
> Un codice di stato 2XX indica l'esito positivo e non verrà restituito alcun errore. Tutti gli altri codici di stato o le eccezioni generate genereranno un errore.

Per ulteriori informazioni, consulta la pagina:

- Concetti di Amazon API Gateway nella API Gateway Developer Guide.
- Politiche IAM per Amazon API Gateway
- Un progetto di esempio che mostra come Effettuare una chiamata a API Gateway

Concetti di Amazon API Gateway nella API Gateway Developer Guide.

Chiama Athena con Step Functions

Step Functions può controllare determinati AWS servizi direttamente da Amazon States Language (ASL). Per ulteriori informazioni, consultare Uso di altri servizi e Passa i parametri a un'API di servizio.

> In che modo l'integrazione Optimized Athena è diversa dall'integrazione con Athena SDK AWS
>
> - Il modello di Esegui un processo (.sync) integrazione è supportato.
> - Non ci sono ottimizzazioni per il modello di Richiesta e risposta integrazione.
> - Il modello di Attendere un callback con il token dell'attività integrazione non è supportato.

L'integrazione del AWS Step Functions servizio con Amazon Athena consente di utilizzare Step Functions per avviare e interrompere l'esecuzione delle query e ottenere i risultati delle query. Utilizzando Step Functions, puoi eseguire query di dati ad hoc o pianificate e recuperare risultati destinati ai tuoi data lake S3. Athena è un servizio serverless, perciò non occorre installare o gestire alcuna infrastruttura e vengono addebitati solo i costi relativi all'esecuzione delle query.

Per l'integrazione AWS Step Functions con Amazon Athena, utilizzi le API di integrazione del servizio Athena fornite.

Le API di integrazione dei servizi sono le stesse delle API Athena corrispondenti. Non tutte le API supportano tutti i modelli di integrazione, come illustrato nella tabella seguente:

API	Richiesta e risposta	Esegui un processo (.sync)
`StartQueryExecution`	✓	✓
`StopQueryExecution`	✓	
`GetQueryExecution`	✓	
`GetQueryResults`	✓	

API Amazon Athena supportate:

> **Note**
>
> Esiste una quota per la dimensione massima dei dati di input o risultato per un'attività in Step Functions. Ciò limita a 256 KB di dati come stringa con codifica UTF-8 quando si inviano o si ricevono dati da un altro servizio. Per informazioni, consulta Quote relative alle esecuzioni di macchine a stati.

- `StartQueryExecution`
  - Sintassi della richiesta
  - Parametri supportati:
    - `ClientRequestToken`
    - `ExecutionParameters`
    - `QueryExecutionContext`
    - `QueryString`
    - `ResultConfiguration`
    - `WorkGroup`
  - `Response syntax`
- `StopQueryExecution`
  - Sintassi della richiesta
  - Parametri supportati:
    - `QueryExecutionId`
- `GetQueryExecution`
  - Sintassi della richiesta
  - Parametri supportati:
    - `QueryExecutionId`
  - `Response syntax`
- `GetQueryResults`
  - Sintassi della richiesta
  - Parametri supportati:
    - `MaxResults`
    - `NextToken`

    - QueryExecutionId
- Response syntax

Quanto segue include uno stato Task che avvia una query 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"
}
```

Per informazioni su come configurare le IAM autorizzazioni quando vengono utilizzate Step Functions con altri AWS servizi, consulta. Politiche IAM per servizi integrati

Gestisci AWS Batch con Step Functions

Step Functions può controllare determinati AWS servizi direttamente da Amazon States Language (ASL). Per ulteriori informazioni, consultare Uso di altri servizi e Passa i parametri a un'API di servizio.

> In che modo l' AWS Batch integrazione ottimizzata è diversa dall'integrazione AWS BatchAWS SDK
>
> - Il modello di Esegui un processo (.sync) integrazione è disponibile.
>
> Nota che non ci sono ottimizzazioni per i modelli Richiesta e risposta di Attendere un callback con il token dell'attività integrazione.

AWS Batch API supportate:

- SubmitJob

- Sintassi della richiesta
- Parametri supportati:
  - `ArrayProperties`
  - `ContainerOverrides`
  - `DependsOn`
  - `JobDefinition`
  - `JobName`
  - `JobQueue`
  - `Parameters`
  - `RetryStrategy`
  - `Timeout`
  - `Tags`
- Sintassi della risposta

> **I parametri in Step Functions sono espressi in PascalCase**
>
> Anche se l'API del servizio nativo è in CamelCase, ad esempio l'`startSyncExecution`azione API, si specificano parametri PascalCase in, ad esempio:. `StateMachineArn`

Quanto segue include uno `Task` stato che invia un AWS Batch processo e attende che venga completato.

```
{
  "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
    }
  }
}
```

Per informazioni su come configurare le IAM autorizzazioni quando vengono utilizzate Step Functions con altri AWS servizi, vedere. Politiche IAM per servizi integrati

Chiama Amazon Bedrock con Step Functions

Step Functions può controllare determinati AWS servizi direttamente da Amazon States Language (ASL). Per ulteriori informazioni, consultare Uso di altri servizi e Passa i parametri a un'API di servizio.

Argomenti

- Amazon BedrockAPI di integrazione dei servizi
- Definizione dello stato delle attività per l'integrazione Amazon Bedrock

Amazon BedrockAPI di integrazione dei servizi

Per l'integrazione AWS Step Functions conAmazon Bedrock, puoi utilizzare le seguenti API. Queste API sono simili alle Amazon Bedrock API corrispondenti, con alcune differenze nei campi di richiesta che vengono passati.

Nella tabella seguente sono descritte le differenze tra ogni API di integrazione del servizio e la relativa API Amazon Bedrock corrispondente:

Amazon BedrockAPI di integrazione dei servizi e API corrispondenti Amazon Bedrock

Amazon BedrockAPI di integrazione dei servizi	Amazon BedrockAPI corrispondente	Differenze
InvokeModel Richiama il Amazon Bedrock modello specificato per eseguire l'inferenza utilizzando l'input fornito nel corpo della richiesta. Si utilizza `InvokeModel` per eseguire l'inferenza per modelli di testo, modelli di immagini e modelli di incorporamento.	InvokeModel	Il corpo della richiesta dell'API di integrazione del Amazon Bedrock servizio include i seguenti parametri aggiuntivi. • `Body`— specifica i dati di input nel formato specificato nell'intestazione della richiesta del tipo di contenuto. `Body`contiene parametri specifici del modello di destinazione. Se si utilizza l'`InvokeModel` API, è necessario specificare il `Body` parametro. Step Functions non convalida l'input fornito. `Body` Quando si specifica l'`Body`utilizzo dell'integrazione Amazon Bedrock ottimizzata, è possibile specificare un payload fino a 256 KB. Se il payload supera i 256 KB, si consiglia di utilizzarlo. `Input` • `Input`— Speciifica la fonte da cui recuperare i dati di input. Questo campo opzionale è specifico per l'integrazione Amazon

Amazon BedrockAPI di integrazione dei servizi	Amazon BedrockAPI corrispondente	Differenze
		Bedrock ottimizzata con. Step Functions In questo campo è possibile specificare un`S3Uri`. È possibile specificare `Body` nei Parametri `oInput`, ma non entrambi. Quando si specifica `Input` senza specificare un`ContentType` , il tipo di contenuto dell'origine dati di input diventa il valore per`ContentType` . • `Output`— Speciifica la destinazione in cui viene scritta la risposta API. Questo campo opzionale è specifico per l'integrazione Amazon Bedrock ottimizzata conStep Functions. In questo campo è possibile specificare un`S3Uri`. Se si specifica questo campo, il corpo della risposta dell'API viene sostituito con un riferimento alla Amazon S3 posizione dell'output originale.

Amazon BedrockAPI di integrazione dei servizi	Amazon BedrockAPI corrispondente	Differenze
		L'esempio seguente mostra la sintassi di InvokeModel API for Amazon Bedrock integration. `{ "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 Crea un lavoro di ottimizzazione per personalizzare un modello base.	CreateModelCustomizationJob	Nessuno
CreateModelCustomizationJob.sync Crea un lavoro di ottimizzazione per personalizzare un modello base.	CreateModelCustomizationJob	Nessuno

Per informazioni su come configurare le IAM autorizzazioni quando vengono utilizzate Step Functions con altri AWS servizi, consulta. Politiche IAM per servizi integrati

Definizione dello stato delle attività per l'integrazione Amazon Bedrock

La seguente definizione dello stato dell'attività mostra come è possibile eseguire l'integrazione con le macchine Amazon Bedrock a stati. Questo esempio mostra uno stato Task che estrae il risultato completo dell'invocazione del modello specificata dal percorso,. `result_one` Questo si basa sui parametri di inferenza per i modelli di base. Questo esempio utilizza il modello di linguaggio di grandi dimensioni (LLM) di Cohere Command.

```
{
  "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**
>
> Per implementare un esempio di macchina a stati che si integra con Amazon Bedrock to your, vedi. Account AWSEsegui il concatenamento dei prompt AI con Amazon Bedrock

Chiama AWS CodeBuild con Step Functions

Step Functions può controllare determinati AWS servizi direttamente da Amazon States Language (ASL). Per ulteriori informazioni, consultare Uso di altri servizi e Passa i parametri a un'API di servizio.

In che modo l' CodeBuild integrazione ottimizzata è diversa dall'integrazione CodeBuild AWS SDK

- Il modello di Esegui un processo (.sync) integrazione è supportato.
- Dopo aver chiamato `StopBuild` or`StopBuildBatch`, il batch di build o build non è immediatamente eliminabile finché non viene completato un lavoro interno CodeBuild per finalizzare lo stato della build o delle build. Se si tenta di utilizzare `BatchDeleteBuilds` o `DeleteBuildBatch` durante questo periodo, è possibile che la build o il batch di compilazione non vengano eliminati. Le integrazioni di servizio ottimizzate per `BatchDeleteBuilds` e `DeleteBuildBatch` includono un nuovo tentativo interno per semplificare il caso d'uso dell'eliminazione immediatamente dopo l'interruzione.

L'integrazione del AWS Step Functions servizio con AWS CodeBuild consente di utilizzare Step Functions per attivare, interrompere e gestire le build e condividere report di build. Utilizzando Step Functions, è possibile progettare ed eseguire pipeline di integrazione continua per la convalida delle modifiche software per le applicazioni.

Non tutte le API supportano tutti i modelli di integrazione, come illustrato nella tabella seguente:

API	Richiesta e risposta	Esegui un processo (.sync)
StartBuild	✓	✓
StopBuild	✓	
BatchDeleteBuilds	✓	
BatchGetReports	✓	
StartBuildBatch	✓	✓
StopBuildBatch	✓	
RetryBuildBatch	✓	✓
DeleteBuildBatch	✓	

> **I parametri in Step Functions sono espressi in PascalCase**
>
> Anche se l'API del servizio nativo è in CamelCase, ad esempio l'`startSyncExecution`azione API, si specificano parametri PascalCase in, ad esempio:. `StateMachineArn`

CodeBuild API e sintassi supportate:

- `StartBuild`
  - Sintassi della richiesta
  - Parametri supportati:
    - `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

- Sintassi della risposta

- StopBuild
  - Sintassi della richiesta
  - Parametri supportati:
    - Id
  - Sintassi della risposta
- BatchDeleteBuilds
  - Sintassi della richiesta
  - Parametri supportati:
    - Ids
  - Sintassi della risposta
- BatchGetReports
  - Sintassi della richiesta
  - Parametri supportati:
    - ReportArns
  - Sintassi della risposta
- StartBuildBatch
  - Sintassi della richiesta
  - Parametri supportati:
    - 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

- Sintassi della risposta

- StopBuildBatch
  - Sintassi della richiesta
  - Parametri supportati:
    - Id
  - Sintassi della risposta
- RetryBuildBatch
  - Sintassi della richiesta
  - Parametri supportati:
    - Id
    - IdempotencyToken
    - RetryType
  - Sintassi della risposta
- DeleteBuildBatch
  - Sintassi della richiesta
  - Parametri supportati:
    - Id
  - Sintassi della risposta

> **Note**
>
> Puoi utilizzare l'operatore di discesa ricorsiva JSONPath (`..`) per `BatchDeleteBuilds`. Questo restituisce una matrice e consente di trasformare il campo `Arn` da `StartBuild` in un parametro `Ids` plurale, come illustrato nell'esempio seguente.

```
"BatchDeleteBuilds": {
    "Type": "Task",
    "Resource": "arn:aws:states:::codebuild:batchDeleteBuilds",
    "Parameters": {
        "Ids.$": "$.Build..Arn"
    },
    "Next": "MyNextState"
},
```

Per informazioni su come configurare le IAM autorizzazioni quando vengono utilizzate Step Functions con altri AWS servizi, consulta. Politiche IAM per servizi integrati

Chiama le API DynamoDB con Step Functions

Step Functions può controllare determinati AWS servizi direttamente da Amazon States Language (ASL). Per ulteriori informazioni, consultare Uso di altri servizi e Passa i parametri a un'API di servizio.

> **Note**
>
> Esiste una quota per la dimensione massima dei dati di input o risultato per un'attività in Step Functions. Ciò limita a 256 KB di dati come stringa con codifica UTF-8 quando si inviano o si ricevono dati da un altro servizio. Per informazioni, consulta Quote relative alle esecuzioni di macchine a stati.

> In che modo l'integrazione ottimizzata di DynamoDB è diversa dall'integrazione DynamoDB SDK AWS
>
> - Non esiste alcuna ottimizzazione per il modello di integrazione. Richiesta e risposta
> - Il modello di Attendere un callback con il token dell'attività integrazione non è supportato.
> - Solo `GetItem`, `PutItemUpdateItem`, e le azioni `DeleteItem`API sono disponibili tramite l'integrazione ottimizzata. Altre azioni API, ad esempio, `CreateTable`sono disponibili utilizzando l'integrazione con DynamoDB AWS SDK.

API e sintassi di Amazon DynamoDB supportate:

- `GetItem`
  - Sintassi della richiesta
  - Parametri supportati:
    - `Key`
    - `TableName`
    - `AttributesToGet`
    - `ConsistentRead`

    - ExpressionAttributeNames
    - ProjectionExpression
    - ReturnConsumedCapacity
  - Sintassi della risposta
- PutItem
  - Sintassi della richiesta
  - Parametri supportati:
    - Item
    - TableName
    - ConditionalOperator
    - ConditionExpression
    - Expected
    - ExpressionAttributeNames
    - ExpressionAttributeValues
    - ReturnConsumedCapacity
    - ReturnItemCollectionMetrics
    - ReturnValues
  - Sintassi della risposta
- DeleteItem
  - Sintassi della richiesta
  - Parametri supportati:
    - Key
    - TableName
    - ConditionalOperator
    - ConditionExpression
    - Expected
    - ExpressionAttributeNames
    - ExpressionAttributeValues
    - ReturnConsumedCapacity
    - ReturnItemCollectionMetrics

    - ReturnValues
  - Sintassi della risposta
- UpdateItem
  - Sintassi della richiesta
  - Parametri supportati:
    - Key
    - TableName
    - AttributeUpdates
    - ConditionalOperator
    - ConditionExpression
    - Expected
    - ExpressionAttributeNames
    - ExpressionAttributeValues
    - ReturnConsumedCapacity
    - ReturnItemCollectionMetrics
    - ReturnValues
    - UpdateExpression
  - Sintassi della risposta

> I parametri in sono espressi in Step Functions PascalCase
>
> Anche se l'API del servizio nativo è in CamelCase, ad esempio l'`startSyncExecution`azione API, si specificano parametri PascalCase in, ad esempio:. `StateMachineArn`

Di seguito è riportato uno `Task` stato che recupera un messaggio da 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"
    },
```

Per visualizzare questo stato in un esempio pratico, consultare il progetto di esempio Trasferisci record di dati (Lambda,DynamoDB,Amazon SQS).

Per informazioni su come configurare le IAM autorizzazioni quando vengono utilizzate Step Functions con altri AWS servizi, consulta. Politiche IAM per servizi integrati

Gestisci le attività di Amazon ECS o Fargate con Step Functions

Step Functions può controllare determinati AWS servizi direttamente da Amazon States Language (ASL). Per ulteriori informazioni, consultare Uso di altri servizi e Passa i parametri a un'API di servizio.

In che modo l'integrazione ottimizzata di Amazon ECS/Fargate è diversa dall'integrazione Amazon ECS o Fargate SDK AWS

- Il modello di integrazione è supportato. Esegui un processo (.sync)
- `ecs:runTask`può restituire una risposta HTTP 200, ma avere un `Failures` campo non vuoto come segue:
  - Richiesta di risposta: restituisci la risposta e non fallire l'operazione. Ciò equivale a nessuna ottimizzazione.
  - Esegui un Job o Task Token: se viene rilevato un `Failures` campo non vuoto, l'operazione fallisce con un `AmazonECS.Unknown` errore.

API e sintassi Amazon ECS/Fargate supportate:

> I parametri in sono espressi in Step Functions PascalCase
>
> Anche se l'API del servizio nativo è in CamelCase, ad esempio l'`startSyncExecution`azione API, si specificano parametri PascalCase in, ad esempio:. `StateMachineArn`

- `RunTask` avvia un nuovo task utilizzando la definizione di task specificata.
  - Sintassi della richiesta
  - Parametri supportati:
    - `Cluster`
    - `Group`
    - `LaunchType`
    - `NetworkConfiguration`
    - `Overrides`
    - `PlacementConstraints`
    - `PlacementStrategy`
    - `PlatformVersion`
    - `PropagateTags`
    - `TaskDefinition`
    - `EnableExecuteCommand`
  - Sintassi della risposta

Trasferimento di dati a un'attività Amazon ECS

Step Functions può controllare determinati AWS servizi direttamente da Amazon States Language (ASL). Per ulteriori informazioni, consultare Uso di altri servizi e Passa i parametri a un'API di servizio.

Puoi usarlo `overrides` per sovrascrivere il comando predefinito per un contenitore e passare l'input alle tue attività Amazon ECS. Per informazioni, consulta `ContainerOverride`. Nell'esempio, abbiamo passato JsonPath valori `Task` dall'input allo `Task` stato.

Quanto segue include uno `Task` stato che esegue un'attività Amazon ECS e attende che venga completata.

```
{
 "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
    }
  }
}
```

La riga `"Command.$": "$.commands"` in `ContainerOverrides` passa i comandi dall'input di stato al container.

Per l'esempio precedente, ciascuno dei comandi verrà trasferito come sostituzione di container se l'input per l'esecuzione è il seguente.

```
{
  "commands": [
    "test command 1",
    "test command 2",
    "test command 3"
  ]
}
```

Quanto segue include uno `Task` stato che esegue un'attività Amazon ECS e quindi attende la restituzione del token dell'attività. Per informazioni, consulta Attendere un callback con il token dell'attività.

```
{
```

```
   "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
      }
   }
}
```

Per informazioni su come configurare le IAM autorizzazioni quando vengono utilizzate Step Functions con altri AWS servizi, consulta. Politiche IAM per servizi integrati

Chiama Amazon EKS con Step Functions

Step Functions può controllare determinati AWS servizi direttamente da Amazon States Language (ASL). Per ulteriori informazioni, consultare Uso di altri servizi e Passa i parametri a un'API di servizio.

In che modo l'integrazione ottimizzata di Amazon EKS è diversa dall'integrazione con Amazon EKS AWS SDK

- Il modello di Esegui un processo (.sync) integrazione è supportato.

- Non ci sono ottimizzazioni per il modello di Richiesta e risposta integrazione.
- Il modello di Attendere un callback con il token dell'attività integrazione non è supportato.

Per informazioni su come configurare IAM le autorizzazioni quando vengono utilizzate Step Functions con altri AWS servizi, vedereIAM Policies for Integrated Services.

Step Functions fornisce due tipi di API di integrazione dei servizi per l'integrazione con Amazon Elastic Kubernetes Service. One ti consente di utilizzare le API di Amazon EKS per creare e gestire un cluster Amazon EKS. L'altro ti consente di interagire con il cluster utilizzando l'API Kubernetes ed eseguire lavori come parte del flusso di lavoro dell'applicazione. Puoi utilizzare le integrazioni API Kubernetes con i cluster Amazon EKS creati con Step Functions, con i cluster Amazon EKS creati dallo strumento eksctl o dalla console Amazon EKS o metodi simili. Per ulteriori informazioni, consulta Creazione di un cluster Amazon EKS nella Guida per l'utente di Amazon EKS.

> **Note**
>
> L'integrazione Step Functions EKS supporta solo le API Kubernetes con accesso pubblico agli endpoint. Per impostazione predefinita, gli endpoint del server API dei cluster EKS hanno accesso pubblico. Per ulteriori informazioni, consulta Amazon EKS Cluster Endpoint Access Control nella Amazon EKS User Guide.

Step Functions non termina automaticamente un cluster Amazon EKS se l'esecuzione viene interrotta. Se la tua macchina a stati si arresta prima che il cluster Amazon EKS sia terminato, il cluster potrebbe continuare a funzionare a tempo indeterminato e potrebbe comportare costi aggiuntivi. Per evitare ciò, assicurati che tutti i cluster Amazon EKS che crei siano terminati correttamente. Per ulteriori informazioni, consultare:

- Eliminazione di un cluster nella Amazon EKS User Guide.
- Esegui un processo (.sync)in Modelli di integrazione dei servizi.

> **Note**
>
> Esiste una quota per la dimensione massima dei dati di input o risultato per un'attività in Step Functions. Ciò limita a 256 KB di dati come stringa con codifica UTF-8 quando si inviano o si

ricevono dati da un altro servizio. Per informazioni, consulta Quote relative alle esecuzioni di macchine a stati.

Integrazioni API Kubernetes

Step Functions supporta le seguenti API Kubernetes:

RunJob

L'integrazione del `eks:runJob` servizio ti consente di eseguire un job sul tuo cluster Amazon EKS. La `eks:runJob.sync` variante consente di attendere il completamento del processo e, facoltativamente, di recuperare i log.

Il tuo server API Kubernetes deve concedere le autorizzazioni per il ruolo IAM utilizzato dalla tua macchina a stati. Per ulteriori informazioni, consulta Autorizzazioni.

Per il pattern Run a Job (`.sync`), lo stato del lavoro è determinato dal polling. Step Functions effettua inizialmente sondaggi a una velocità di circa 1 sondaggio al minuto. Questa frequenza alla fine rallenta a circa 1 sondaggio ogni 5 minuti. Se hai bisogno di sondaggi più frequenti o hai bisogno di un maggiore controllo sulla strategia di polling, puoi utilizzare l'`eks:call`integrazione per interrogare lo stato del lavoro.

L'`eks:runJob`integrazione è specifica per `batch/v1` Kubernetes Jobs. Per ulteriori informazioni, consulta Jobs nella documentazione di Kubernetes. Se desideri gestire altre risorse Kubernetes, incluse le risorse personalizzate, utilizza l'integrazione del servizio. `eks:call` È possibile utilizzare Step Functions per creare cicli di polling, come dimostrato nel progetto di the section called “Sondaggio per Job Status (AWS Batch Lambda,) ” esempio.

I parametri supportati includono:

- `ClusterName`: il nome del cluster Amazon EKS che desideri chiamare.
  - `Type`: `String`
  - Obbligatorio: sì
- `CertificateAuthority`: I dati del certificato con codifica Base64 necessari per comunicare con il cluster. Puoi ottenere questo valore dalla console Amazon EKS o utilizzando l'DescribeClusterAPI Amazon EKS.
  - `Type`: `String`
  - Obbligatorio: sì

- `Endpoint`: L'URL dell'endpoint per il tuo server API Kubernetes. Puoi ottenere questo valore dalla console Amazon EKS o utilizzando l'DescribeClusterAPI Amazon EKS.
  - `Type`: `String`
  - Obbligatorio: sì
- `Namespace`: lo spazio dei nomi in cui eseguire il processo. Se non viene fornito, viene utilizzato lo spazio dei nomi`default`.
  - `Type`: `String`
  - Obbligatorio: no
- `Job`: La definizione di Kubernetes Job. Vedi Jobs nella documentazione di Kubernetes.
  - `Type`: `JSON` o `String`
  - Obbligatorio: sì
- `LogOptions`: Un insieme di opzioni per controllare il recupero opzionale dei log. Applicabile solo se si utilizza il modello di integrazione del servizio Run a Job (.sync) per attendere il completamento del processo.
  - `Type`: `JSON`
  - Obbligatorio: no
  - I log sono inclusi nella risposta sotto la chiave. `logs` All'interno del job possono essere presenti più pod, ciascuno con più contenitori.

```
{
  ...
  "logs": {
    "pods": {
      "pod1": {
        "containers": {
          "container1": {
            "log": <log>
          },
          ...
        }
      },
      ...
    }
  }
```

  - Il recupero dei log viene eseguito con la massima diligenza possibile. Se si verifica un errore durante il recupero di un registro, al posto del `log` campo ci saranno i campi e. `error` `cause`

- `LogOptions.RetrieveLogs`: Abilita il recupero dei log dopo il completamento del processo. Per impostazione predefinita, i log non vengono recuperati.
  - `Type: Boolean`
  - Obbligatorio: no
- `LogOptions.RawLogs`: Se `RawLogs` è impostato su true, i log verranno restituiti come stringhe non elaborate senza tentare di analizzarli in JSON. Per impostazione predefinita, i log vengono deserializzati in JSON, se possibile. In alcuni casi tale analisi può introdurre modifiche indesiderate, come la limitazione della precisione dei numeri contenenti molte cifre.
  - `Type: Boolean`
  - Obbligatorio: no
- `LogOptions.LogParameters`: L'API Read Log dell'API Kubernetes supporta i parametri di query per controllare il recupero dei log. Ad esempio, è possibile utilizzare `tailLines` o `limitBytes` limitare la dimensione dei log recuperati e rimanere entro la quota di dimensione dei dati di Step Functions. Per ulteriori informazioni, consulta la sezione Read Log del Kubernetes API Reference.
  - `Type: Mappa di String List of Strings`
  - Obbligatorio: no
  - Esempio:

```
"LogParameters": {
  "tailLines": [ "6" ]
}
```

L'esempio seguente include uno `Task` stato che esegue un processo, attende il suo completamento, quindi recupera i log del processo:

```
{
  "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`

L'integrazione dei `eks:call` servizi consente di utilizzare l'API Kubernetes per leggere e scrivere oggetti di risorse Kubernetes tramite un endpoint dell'API Kubernetes.

Il tuo server API Kubernetes deve concedere le autorizzazioni per il ruolo IAM utilizzato dalla tua macchina a stati. Per ulteriori informazioni, consulta Autorizzazioni.

Per ulteriori informazioni sulle operazioni disponibili, consulta il Kubernetes API Reference.

I parametri supportati includono: `Call`

- `ClusterName`: il nome del cluster Amazon EKS che desideri chiamare.
  - `Type`: Stringa
  - Campo obbligatorio: sì
- `CertificateAuthority`: I dati del certificato con codifica Base64 necessari per comunicare con il cluster. Puoi ottenere questo valore dalla console Amazon EKS o utilizzando l'DescribeClusterAPI Amazon EKS.
  - `Type`: `String`
  - Campo obbligatorio: sì
- `Endpoint`: L'URL dell'endpoint per il tuo server API Kubernetes. Puoi trovare questo valore sulla console Amazon EKS o utilizzando l' DescribeCluster API di Amazon EKS.
  - `Type`: `String`
  - Campo obbligatorio: sì
- `Method`: il metodo HTTP della tua richiesta. Uno tra: `GET`, `POST`, `PUT`, `DELETE`, `HEAD`, o `PATCH`.
  - `Type`: `String`
  - Campo obbligatorio: sì
- `Path`: Il percorso HTTP dell'operazione dell'API REST di Kubernetes.
  - `Type`: `String`
  - Campo obbligatorio: sì
- `QueryParameters`: i parametri di query HTTP dell'operazione dell'API REST di Kubernetes.
  - `Type:` Mappa di `String` `List of Strings`
  - Campo obbligatorio: no
  - Esempio:

```
"QueryParameters": {
  "labelSelector": [ "job-name=example-job" ]
}
```

- `RequestBody`: il corpo del messaggio HTTP dell'operazione dell'API REST di Kubernetes.

- Type: `JSON` o `String`
- Campo obbligatorio: no

Quanto segue include uno `Task` stato che utilizza `eks:call` per elencare i pod appartenenti al lavoro. `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
    }
  }
}
```

Quanto segue include uno `Task` stato che utilizza `eks:call` per eliminare il lavoro `example-job` e lo imposta `propagationPolicy` per garantire che vengano eliminati anche i pod del lavoro.

```
{
  "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 supportate

Le API e la sintassi di Amazon EKS supportate includono:

- `CreateCluster`
  - Sintassi della richiesta
  - Sintassi della risposta

    Quando un cluster Amazon EKS viene creato utilizzando l'integrazione del `eks:createCluster` servizio, il ruolo IAM viene aggiunto alla tabella di autorizzazione RBAC di Kubernetes come amministratore (con autorizzazioni system:masters). Inizialmente, solo quell'entità IAM può effettuare chiamate al server API Kubernetes. Per ulteriori informazioni, consultare:

    - Gestione degli utenti o dei ruoli IAM per il tuo cluster nella Amazon EKS User Guide
    - La Autorizzazioni sezione

    Amazon EKS utilizza ruoli collegati ai servizi che contengono le autorizzazioni richieste da Amazon EKS per chiamare altri servizi per tuo conto. Se questi ruoli collegati ai servizi non esistono già nel tuo account, devi aggiungere l'`iam:CreateServiceLinkedRole`autorizzazione al ruolo IAM utilizzato da Step Functions. Per ulteriori informazioni, consulta Using Service-Linked Roles nella Amazon EKS User Guide.

    Il ruolo IAM utilizzato da Step Functions deve disporre `iam:PassRole` delle autorizzazioni per passare il ruolo IAM del cluster ad Amazon EKS. Per ulteriori informazioni, consulta il ruolo IAM del cluster Amazon EKS nella Guida per l'utente di Amazon EKS.

- `DeleteCluster`
  - Sintassi della richiesta
  - Sintassi della risposta

  È necessario eliminare qualsiasi profilo o gruppo di nodi Fargate prima di eliminare un cluster.
- `CreateFargateProfile`
  - Sintassi della richiesta
  - Sintassi della risposta

  Amazon EKS utilizza ruoli collegati ai servizi che contengono le autorizzazioni richieste da Amazon EKS per chiamare altri servizi per tuo conto. Se questi ruoli collegati ai servizi non esistono già nel tuo account, devi aggiungere l'`iam:CreateServiceLinkedRole`autorizzazione al ruolo IAM utilizzato da Step Functions. Per ulteriori informazioni, consulta Using Service-Linked Roles nella Amazon EKS User Guide.

  Amazon EKS su Fargate potrebbe non essere disponibile in tutte le regioni. Per informazioni sulla disponibilità delle regioni, consulta la sezione Fargate nella Guida per l'utente di Amazon EKS.

  Il ruolo IAM utilizzato da Step Functions deve disporre `iam:PassRole` delle autorizzazioni per passare il ruolo IAM di esecuzione del pod ad Amazon EKS. Per ulteriori informazioni, consulta il ruolo di esecuzione del Pod nella Guida per l'utente di Amazon EKS.
- `DeleteFargateProfile`
  - Sintassi della richiesta
  - Sintassi della risposta
- `CreateNodegroup`
  - Sintassi della richiesta
  - Sintassi della risposta

  Amazon EKS utilizza ruoli collegati ai servizi che contengono le autorizzazioni richieste da Amazon EKS per chiamare altri servizi per tuo conto. Se questi ruoli collegati ai servizi non esistono già nel tuo account, devi aggiungere l'`iam:CreateServiceLinkedRole`autorizzazione al ruolo IAM utilizzato da Step Functions. Per ulteriori informazioni, consulta Using Service-Linked Roles nella Amazon EKS User Guide.

Il ruolo IAM utilizzato da Step Functions deve disporre `iam:PassRole` delle autorizzazioni per passare il ruolo IAM del nodo ad Amazon EKS. Per ulteriori informazioni, consulta Using Service-Linked Roles nella Amazon EKS User Guide.

- `DeleteNodegroup`
  - Sintassi della richiesta
  - Sintassi della risposta

Quanto segue include un `Task` che crea un cluster 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
    }
  }
}
```

Quanto segue include uno `Task` stato che elimina un cluster Amazon EKS.

```
{
  "StartAt": "DeleteCluster.sync",
  "States": {
    "DeleteCluster.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:deleteCluster.sync",
      "Parameters": {
        "Name": "MyCluster"
```

```
      },
      "End": true
    }
  }
}
```

Quanto segue include uno `Task` stato che crea un profilo 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
    }
  }
}
```

Quanto segue include uno `Task` stato che elimina un profilo Fargate.

```
{
  "StartAt": "DeleteFargateProfile.sync",
  "States": {
    "DeleteFargateProfile.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:deleteFargateProfile.sync",
      "Parameters": {
        "ClusterName": "MyCluster",
        "FargateProfileName": "MyFargateProfile"
      },
      "End": true
    }
```

```
  }
}
```

Quanto segue include uno `Task` stato che crea un gruppo di nodi.

```
{
  "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
    }
  }
}
```

Quanto segue include uno `Task` stato che elimina un gruppo di nodi.

```
{
  "StartAt": "DeleteNodegroup.sync",
  "States": {
    "DeleteNodegroup.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:deleteNodegroup.sync",
      "Parameters": {
        "ClusterName": "MyCluster",
        "NodegroupName": "MyNodegroup"
      },
      "End": true
    }
  }
}
```

Autorizzazioni

Quando un cluster Amazon EKS viene creato utilizzando l'integrazione del `eks:createCluster` servizio, il ruolo IAM viene aggiunto alla tabella di autorizzazione RBAC di Kubernetes come amministratore, con autorizzazioni. `system:masters` Inizialmente, solo quell'entità IAM può effettuare chiamate al server API Kubernetes. Ad esempio, non sarai in grado di utilizzare kubectl per interagire con il tuo server API Kubernetes, a meno che non assumi lo stesso ruolo della tua macchina a stati Step Functions o se configuri Kubernetes per concedere autorizzazioni a entità IAM aggiuntive. Per ulteriori informazioni, consulta la sezione Gestione degli utenti o dei ruoli IAM per il tuo cluster nella Amazon EKS User Guide.

Puoi aggiungere l'autorizzazione per altre entità IAM, come utenti o ruoli, aggiungendole allo spazio dei nomi `aws-auth` `ConfigMap` nel sistema kube. Se stai creando il tuo cluster da Step Functions, utilizza l'integrazione del `eks:call` servizio.

Quanto segue include uno `Task` stato che crea `aws-auth` `ConfigMap` e concede l'`system:masters`autorizzazione all'utente `arn:aws:iam::123456789012:user/my-user` e al ruolo `arn:aws:iam::123456789012:role/my-role` IAM.

```
{
  "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**
>
> Potresti vedere l'ARN per un ruolo IAM visualizzato in un formato che include il percorso / service-role/, ad esempio. `arn:aws:iam::123456789012:role/service-role/my-role` Questo token del percorso del ruolo di servizio non deve essere incluso nell'elenco del ruolo in. `aws-auth`

Quando il cluster viene creato per la prima volta, non `aws-auth` `ConfigMap` esisterà, ma verrà aggiunto automaticamente se si crea un profilo Fargate. È possibile recuperare il valore corrente di`aws-auth`, aggiungere le autorizzazioni aggiuntive e PUT una nuova versione. Di solito è più facile da creare `aws-auth` prima del profilo Fargate.

Se il tuo cluster è stato creato al di fuori di Step Functions, puoi configurare kubectl per comunicare con il tuo server API Kubernetes. Quindi, creane un nuovo `aws-auth` `ConfigMap` utilizzo `kubectl apply -f aws-auth.yaml` o modificane uno già esistente utilizzando. `kubectl edit -n kube-system configmap/aws-auth` Per ulteriori informazioni, consultare:

- Crea un kubeconfig per Amazon EKS nella Amazon EKS User Guide.
- Gestione degli utenti o dei ruoli IAM per il tuo cluster nella Amazon EKS User Guide.

Se il tuo ruolo IAM non dispone di autorizzazioni sufficienti in Kubernetes, le integrazioni `eks:call` o di `eks:runJob` servizio falliranno con il seguente errore:

```
Error:
EKS.401

Cause:
{
  "ResponseBody": {
```

```
    "kind": "Status",
    "apiVersion": "v1",
    "metadata": {},
    "status": "Failure",
    "message": "Unauthorized",
    "reason": "Unauthorized",
    "code": 401
  },
  "StatusCode": 401,
  "StatusText": "Unauthorized"
}
```

Chiama Amazon EMR con Step Functions

Step Functions può controllare determinati AWS servizi direttamente da Amazon States Language (ASL). Per ulteriori informazioni, consultare Uso di altri servizi e Passa i parametri a un'API di servizio.

> **In che modo l'integrazione ottimizzata di Amazon EMR è diversa dall'integrazione con Amazon EMR SDK AWS**
>
> L'integrazione ottimizzata del servizio Amazon EMR include un set personalizzato di API che racchiudono le API Amazon EMR sottostanti, descritte di seguito. Per questo motivo, si differenzia in modo significativo dall'integrazione del servizio SDK Amazon AWS EMR. Inoltre, è supportato il modello di Esegui un processo (.sync) integrazione.

Per l'integrazione AWS Step Functions con Amazon EMR, utilizzi le API di integrazione del servizio Amazon EMR fornite. Le API di integrazione dei servizi sono simili alle corrispondenti API di Amazon EMR, con alcune differenze nei campi che vengono passati e nelle risposte restituite.

Step Functions non termina automaticamente un cluster Amazon EMR se l'esecuzione viene interrotta. Se la tua macchina a stati si arresta prima che il cluster Amazon EMR sia terminato, il cluster potrebbe continuare a funzionare a tempo indeterminato e potrebbe comportare costi aggiuntivi. Per evitare ciò, assicurati che tutti i cluster Amazon EMR che crei siano terminati correttamente. Per ulteriori informazioni, consultare:

- Controlla la terminazione del cluster nella Guida per l'utente di Amazon EMR.

- La sezione Service Integration Patterns. Esegui un processo (.sync)

> **Note**
>
> A partire da `emr-5.28.0` ora, puoi specificare il parametro `StepConcurrencyLevel` durante la creazione di un cluster per consentire l'esecuzione di più passaggi in parallelo su un singolo cluster. È possibile utilizzare Step Functions `Map` e `Parallel` gli stati per inviare il lavoro in parallelo al cluster.

La disponibilità dell'integrazione del servizio Amazon EMR è soggetta alla disponibilità delle API di Amazon EMR. Consulta la documentazione di Amazon EMR per le limitazioni nelle regioni speciali.

> **Note**
>
> Per l'integrazione con Amazon EMR, Step Functions utilizza una frequenza di polling dei job codificata di 60 secondi per i primi 10 minuti e per i 300 secondi successivi.

La tabella seguente descrive le differenze tra ogni API di integrazione dei servizi e la corrispondente API Amazon EMR.

API di integrazione del servizio Amazon EMR e API Amazon EMR corrispondenti

API di integrazione dei servizi Amazon EMR	API EMR corrispondente	Differenze
createCluster Crea e avvia l'esecuzione di un cluster (flusso di lavoro). Amazon EMR è collegato direttamente a un tipo unico di ruolo IAM noto come ruolo collegato ai servizi. Perché `createCluster` e `createCluster.sync` funzionino, è necessario	runJobFlow	`createCluster` utilizza la stessa sintassi di richiesta di runJobFlow, ad eccezione di quanto segue: • Il campo `Instances.KeepJobFlowAliveWhenNoSteps` è obbligatorio e deve avere il valore booleano `TRUE`.

API di integrazione dei servizi Amazon EMR	API EMR corrispondente	Differenze

aver configurato le autorizzazioni necessarie per creare il ruolo collegato al servizio `AWSServiceRoleForEMRCleanup`. Per ulteriori informazioni su questo argomento, inclusa una dichiarazione che puoi aggiungere alla tua politica di autorizzazioni IAM, consulta Using the Service-Linked Role for Amazon EMR.		• Il campo `Steps` non è consentito.
		• Deve essere fornito il campo `Instances.InstanceFleets[index].Name`, che deve essere univoco se viene utilizzata l'API del connettore `modifyInstanceFleetByName` opzionale.
		• Deve essere fornito il campo `Instances.InstanceGroups[index].Name`, che deve essere univoco se viene utilizzata l'API `modifyInstanceGroupByName` opzionale.
		
		La risposta è:
		
		`{ "ClusterId": "string" }`
		
		Amazon EMR utilizza questo:
		
		`{ "JobFlowId": "string" }`

API di integrazione dei servizi Amazon EMR	API EMR corrispondente	Differenze
createCluster.sync Crea e avvia l'esecuzione di un cluster (flusso di lavoro).	runJobFlow	Lo stesso di `createCluster` , ma attende che il cluster raggiunga lo stato `WAITING`.

setClusterTerminationProtezione
 	
Blocca un cluster (flusso di lavoro) in modo che le istanze EC2 nel cluster non possano essere terminate dall'intervento dell'utente, da una chiamata API o in caso di errore del flusso di lavoro.
	setTerminationProtection	La richiesta utilizza questo:
		
		`{ "ClusterId": "string" }`
		
		Amazon EMR utilizza questo:
		
		`{ "JobFlowIds": ["string"] }`

terminateCluster
 	
Arresta un cluster (flusso di lavoro).
	terminateJobFlows	La richiesta utilizza questo:
		
		`{ "ClusterId": "string" }`
		
		Amazon EMR utilizza questo:
		
		`{ "JobFlowIds": ["string"] }`

terminateCluster.sync Arresta un cluster (flusso di lavoro).	terminateJobFlows	Lo stesso di `terminateCluster` , ma attende che il cluster venga terminato.

API di integrazione dei servizi Amazon EMR	API EMR corrispondente	Differenze

addStep
 	
Aggiunge un nuovo passaggio a un cluster in esecuzione.
 	
Facoltativamente, puoi anche specificare il `ExecutionRoleArn` parametro durante l'utilizzo di questa API.
	addJobFlowPassi	La richiesta utilizza la chiave`"ClusterId"` . Usi di Amazon EMR. `"JobFlowId"` La richiesta utilizza un singolo passaggio.
		
		`{ "Step": <"StepConfig object"> }`
		
		Amazon EMR utilizza questo:
		
		`{ "Steps": [<StepConfig objects>] }`
		
		La risposta è:
		
		`{ "StepId": "string" }`
		
		Amazon EMR restituisce questo:
		
		`{ "StepIds": [<strings>] }`

API di integrazione dei servizi Amazon EMR	API EMR corrispondente	Differenze
addStep.sync Aggiunge un nuovo passaggio a un cluster in esecuzione. Facoltativamente, puoi anche specificare il `ExecutionRoleArn` parametro durante l'utilizzo di questa API.	addJobFlowPassi	Come `addStep`, ma attende il completamento del passaggio.

API di integrazione dei servizi Amazon EMR	API EMR corrispondente	Differenze

cancelStep
 	
Annulla un passaggio in sospeso in un cluster in esecuzione.
	cancelSteps	La richiesta utilizza questo:
		
		`{ "StepId": "string" }`
		
		Amazon EMR utilizza questo:
		
		`{ "StepIds": [<strings >] }`
		
		La risposta è:
		
		`{ "CancelStepsInfo": <CancelStepsInfo object> }`
		
		Amazon EMR utilizza questo:
		
		`{ "CancelStepsInfoList": [<CancelStepsInfo objects>] }`

API di integrazione dei servizi Amazon EMR	API EMR corrispondente	Differenze
modifyInstanceFleetByName Modifica le capacità Spot target e on demand per il parco di istanze con il `InstanceFleetName` specificato.	modifyInstanceFleet	La richiesta è la stessa di quella di `modifyInstanceFleet`, ad eccezione di quanto segue: • Il campo `Instance.InstanceFleetId` non è consentito. • In fase di esecuzione `InstanceFleetId` viene determinato automaticamente dall'integrazione del servizio chiamando `ListInstanceFleets` e analizzando il risultato.

API di integrazione dei servizi Amazon EMR	API EMR corrispondente	Differenze

modifyInstanceGroupByName
 	
Modifica il numero di nodi e le impostazioni di configurazione di un gruppo di istanze.
	modifyInstanceGroups	La richiesta è questa:
		
		`{ "ClusterId": "string", "InstanceGroup": <InstanceGroupModifyConfig object> }`
		
		Amazon EMR utilizza un elenco:
		
		`{ "ClusterId": ["string"], "InstanceGroups": [<InstanceGroupModifyConfig objects>] }`
		
		All'interno dell'oggetto `InstanceGroupModifyConfig` il campo `InstanceGroupId` non è consentito.
		
		È stato aggiunto un nuovo campo, `InstanceGroupName`. In fase di esecuzione `InstanceGroupId` viene determinato automaticamente dall'integrazione del servizio chiamando `ListInsta`

API di integrazione dei servizi Amazon EMR	API EMR corrispondente	Differenze
		nceGroups e analizzando il risultato.

L'esempio seguente include uno stato `Task` che crea un cluster.

```
"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
}
```

Di seguito è riportato uno stato `Task` che consente la protezione di terminazione.

```
"Enable_Termination_Protection": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:setClusterTerminationProtection",
    "Parameters": {
        "ClusterId.$": "$.ClusterId",
        "TerminationProtected": true
    },
    "End": true
}
```

Di seguito è riportato uno stato `Task` che invia un passaggio a un cluster.

```
"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
}
```

Di seguito è riportato uno stato `Task` che annulla un passaggio.

```
"Cancel_Step_One": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:cancelStep",
    "Parameters": {
        "ClusterId.$": "$.ClusterId",
        "StepId.$": "$.AddStepsResult.StepId"
    },
    "End": true
}
```

Di seguito è riportato uno stato `Task` che termina un cluster.

```
"Terminate_Cluster": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:terminateCluster.sync",
    "Parameters": {
        "ClusterId.$": "$.ClusterId"
    },
    "End": true
}
```

Di seguito è riportato uno stato `Task` che consente di scalare un cluster verso l'alto o verso il basso per un gruppo di istanze.

```
"ModifyInstanceGroupByName": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:modifyInstanceGroupByName",
    "Parameters": {
```

```
        "ClusterId": "j-1234567890123",
        "InstanceGroupName": "MyCoreGroup",
        "InstanceGroup": {
            "InstanceCount": 8
        }
    },
    "End": true
}
```

Di seguito è riportato uno stato `Task` che consente di scalare un cluster verso l'alto o verso il basso per un parco di istanze.

```
"ModifyInstanceFleetByName": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:modifyInstanceFleetByName",
    "Parameters": {
        "ClusterId": "j-1234567890123",
        "InstanceFleetName": "MyCoreFleet",
        "InstanceFleet": {
            "TargetOnDemandCapacity": 8,
            "TargetSpotCapacity": 0
        }
    },
    "End": true
}
```

Per informazioni su come configurare le IAM autorizzazioni quando vengono utilizzate Step Functions con altri AWS servizi, consulta. Politiche IAM per servizi integrati

Chiama Amazon EMR su EKS con AWS Step Functions

Step Functions può controllare determinati AWS servizi direttamente da Amazon States Language (ASL). Per ulteriori informazioni, consultare Uso di altri servizi e Passa i parametri a un'API di servizio.

> In che modo l'integrazione ottimizzata di Amazon EMR su EKS è diversa dall'integrazione Amazon EMR su EKS SDK AWS
>
> - Il modello di Esegui un processo (.sync) integrazione è supportato.
> - Non ci sono ottimizzazioni per il modello di Richiesta e risposta integrazione.

- Il modello di Attendere un callback con il token dell'attività integrazione non è supportato.

> **Note**
>
> Per l'integrazione con Amazon EMR, Step Functions utilizza una frequenza di polling dei job codificata di 60 secondi per i primi 10 minuti e per i 300 secondi successivi.

Per l'integrazione AWS Step Functions con Amazon EMR su EKS, utilizza le API di integrazione dei servizi Amazon EMR on EKS. Le API di integrazione dei servizi sono le stesse delle corrispondenti API Amazon EMR su EKS, ma non tutte le API supportano tutti i modelli di integrazione, come mostrato nella tabella seguente.

API	Richiesta di risposta	Esegui un lavoro (.sync)
CreateVirtualCluster	✓	
DeleteVirtualCluster	✓	✓
StartJobRun	✓	✓

Amazon EMR su API EKS supportate:

> **Note**
>
> Esiste una quota per la dimensione massima dei dati di input o risultato per un'attività in Step Functions. Ciò limita a 256 KB di dati come stringa con codifica UTF-8 quando si inviano o si ricevono dati da un altro servizio. Per informazioni, consulta Quote relative alle esecuzioni di macchine a stati.

- `CreateVirtualCluster`
  - Sintassi della richiesta
  - Parametri supportati
  - Sintassi della risposta

- `DeleteVirtualCluster`
  - Sintassi della richiesta
  - Parametri supportati
  - Sintassi della risposta
- `StartJobRun`
  - Sintassi della richiesta
  - Parametri supportati
  - Sintassi della risposta

Quanto segue include uno `Task` stato che crea un cluster virtuale.

```
"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
}
```

Quanto segue include uno `Task` stato che invia un processo a un cluster virtuale e attende che venga completato.

```
"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
}
```

Quanto segue include uno `Task` stato che elimina un cluster virtuale e attende il completamento dell'eliminazione.

```
"Delete_Virtual_Cluster": {
  "Type": "Task",
  "Resource": "arn:aws:states:::emr-containers:deleteVirtualCluster.sync",
  "Parameters": {
    "Id.$": "$.VirtualClusterId"
  },
  "End": true
}
```

Per informazioni su come configurare le IAM autorizzazioni quando vengono utilizzate Step Functions con altri AWS servizi, vedere. Politiche IAM per servizi integrati

Chiama Amazon EMR Serverless con Step Functions

Step Functions può controllare determinati AWS servizi direttamente da Amazon States Language (ASL). Per ulteriori informazioni, consultare Uso di altri servizi e Passa i parametri a un'API di servizio.

> In che modo l'EMR Serverlessintegrazione ottimizzata è diversa dall'integrazione EMR Serverless AWS SDK
>
> - L'integrazione ottimizzata dei EMR Serverless servizi dispone di un set personalizzato di API che racchiudono le API sottostantiEMR Serverless. Grazie a questa personalizzazione, l'EMR Serverlessintegrazione ottimizzata differisce notevolmente dall'integrazione del EMR Serverless AWS servizio SDK. Inoltre, l'EMR Serverlessintegrazione ottimizzata supporta Esegui un processo (.sync) il modello di integrazione.
> - Il modello di Attendere un callback con il token dell'attività integrazione non è supportato.

In questo argomento

- EMR ServerlessAPI di integrazione dei servizi
- Casi d'uso dell'integrazione EMR Serverless

EMR ServerlessAPI di integrazione dei servizi

Per l'integrazione AWS Step Functions conEMR Serverless, puoi utilizzare le seguenti sei API di integrazione dei EMR Serverless servizi. Queste API di integrazione dei servizi sono simili alle EMR Serverless API corrispondenti, con alcune differenze nei campi passati e nelle risposte restituite.

Nella tabella seguente sono descritte le differenze tra ogni API di integrazione del servizio e la relativa API EMR Serverless corrispondente:

EMR ServerlessAPI di integrazione dei servizi e API corrispondenti EMR Serverless

EMR ServerlessAPI di integrazione dei servizi	EMR ServerlessAPI corrispondente	Differenze
Crea applicazione Crea un'applicazione. EMR Serverlessè collegato a un tipo di IAM ruolo univoco noto come ruolo collegato al servizio. Perché `createApplication` e `createApplication.sync` funzionino, è necessario aver configurato le autorizzazioni necessarie per creare il ruolo collegato al servizio `AWS ServiceRoleForAmazonEMRServerless` . Per ulteriori informazioni su questo argomento, inclusa una dichiarazione che puoi aggiungere alla tua politica di IAM autorizzazioni, vedi Utilizzo di ruoli collegati al servizio per. EMR Serverless	CreateApplication	Nessuno

EMR ServerlessAPI di integrazione dei servizi	EMR ServerlessAPI corrispondente	Differenze
CreateApplication.sync Crea un'applicazione.	CreateApplication	Nessuna differenza tra le richieste e le risposte dell'API e dell'EMR ServerlessAPI di integrazione dei servizi. EMR Serverless Tuttavia, CreateApplication.sync attende che l'applicazione raggiunga lo stato. `CREATED`
Avvia l'applicazione Avvia un'applicazione specificata e inizializza la capacità iniziale dell'applicazione, se configurata.	StartApplication	La risposta dell'EMR ServerlessAPI non contiene dati, ma la risposta dell'API EMR Serverless di integrazione del servizio include i seguenti dati. `{ "ApplicationId": "string" }`
Avvia Application.sync Avvia un'applicazione specificata e inizializza la capacità iniziale, se configurata.	StartApplication	La risposta dell'EMR ServerlessAPI non contiene dati, ma la risposta dell'API EMR Serverless di integrazione del servizio include i seguenti dati. `{ "ApplicationId": "string" }` Inoltre, StartApplication.sync attende che l'applicazione raggiunga lo stato. `STARTED`

EMR ServerlessAPI di integrazione dei servizi	EMR ServerlessAPI corrispondente	Differenze
Arresta l'applicazione Arresta un'applicazione specificata e rilascia la capacità iniziale, se configurata. Tutti i processi pianificati e in esecuzione devono essere completati o annullati prima di interrompere un'applicazione.	StopApplication	La risposta dell'EMR ServerlessAPI non contiene dati, ma la risposta dell'API EMR Serverless di integrazione del servizio include i seguenti dati. `{ "ApplicationId": "string" }`
Arresta Application.sync Arresta un'applicazione specificata e rilascia la capacità iniziale, se configurata. Tutti i processi pianificati e in esecuzione devono essere completati o annullati prima di interrompere un'applicazione.	StopApplication	La risposta dell'EMR ServerlessAPI non contiene dati, ma la risposta dell'API EMR Serverless di integrazione del servizio include i seguenti dati. `{ "ApplicationId": "string" }` Inoltre, StopApplication.sync attende che l'applicazione raggiunga lo stato. `STOPPED`

EMR ServerlessAPI di integrazione dei servizi	EMR ServerlessAPI corrispondente	Differenze
Elimina applicazione Elimina un'applicazione. Un'applicazione deve trovarsi nello `CREATED` stato `STOPPED` o per poter essere eliminata.	DeleteApplication	La risposta dell'EMR ServerlessAPI non contiene dati, ma la risposta dell'API di integrazione del EMR Serverless servizio include i seguenti dati. `{ "ApplicationId": "string" }`
Elimina Application.sync Elimina un'applicazione. Un'applicazione deve trovarsi nello `CREATED` stato `STOPPED` o per poter essere eliminata.	DeleteApplication	La risposta dell'EMR ServerlessAPI non contiene dati, ma la risposta dell'API di integrazione del EMR Serverless servizio include i seguenti dati. `{ "ApplicationId": "string" }` Inoltre, StopApplication.sync attende che l'applicazione raggiunga lo stato. `TERMINATED`
startJobRun Avvia l'esecuzione di un processo.	StartJobRun	Nessuno

EMR ServerlessAPI di integrazione dei servizi	EMR ServerlessAPI corrispondente	Differenze
startJobRun.sync Avvia l'esecuzione di un processo.	StartJobRun	Nessuna differenza tra le richieste e le risposte dell'EMR ServerlessAPI e dell'API EMR Serverless di integrazione dei servizi. Tuttavia, startJobRun.sync attende che l'applicazione raggiunga lo `SUCCESS` stato.
cancelJobRun Annulla l'esecuzione di un processo.	CancelJobRun	Nessuno
cancelJobRun.sync Annulla l'esecuzione di un processo.	CancelJobRun	Nessuna differenza tra le richieste e le risposte dell'EMR ServerlessAPI e dell'API di integrazione dei EMR Serverless servizi. Tuttavia, cancelJobRun.sync attende che l'applicazione raggiunga lo `CANCELLED` stato.

Casi d'uso dell'integrazione EMR Serverless

Per l'integrazione ottimizzata dei EMR Serverless servizi, si consiglia di creare una singola applicazione e quindi di utilizzarla per eseguire più processi. Ad esempio, in una macchina a stato singolo, è possibile includere più startJobRunrichieste, che utilizzano tutte la stessa applicazione. I seguenti esempi di Attività stato mostrano i casi d'uso con Step Functions cui integrare le EMR Serverless API. Per informazioni su altri casi d'uso diEMR Serverless, consulta What is Amazon EMR Serverless.

> **Tip**
>
> Per distribuire un esempio di macchina a stati che si integra EMR Serverless per eseguire più lavori sul tuo computer Account AWS, consulta. Esegui un EMR Serverless lavoro

- Creazione di un'applicazione
- Avviare un'applicazione
- Arresta un'applicazione
- Eliminazione di un'applicazione
- Avviare un processo in un'applicazione
- Annullare un lavoro in un'applicazione

Per informazioni su come configurare le IAM autorizzazioni quando vengono utilizzate Step Functions con altri AWS servizi, vedere. Politiche IAM per servizi integrati

Negli esempi mostrati nei seguenti casi d'uso, sostituisci il testo in *corsivo* con le informazioni specifiche della risorsa. Ad esempio, sostituiscilo *yourApplicationId*con l'ID della tua applicazione, ad esempio. EMR Serverless `00yv7iv71inak893`

Creazione di un'applicazione

Il seguente esempio di stato Task crea un'applicazione utilizzando l'API di integrazione del servizio 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
}
```

Avviare un'applicazione

Il seguente esempio di stato Task avvia un'applicazione utilizzando l'API di integrazione del servizio StartApplication.sync.

```
"Start_Application": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:startApplication.sync",
    "Parameters": {
        "ApplicationId": "yourApplicationId"
    },
    "End": true
}
```

Arresta un'applicazione

Il seguente esempio di stato Task interrompe un'applicazione utilizzando l'API di integrazione del servizio StopApplication.sync.

```
"Stop_Application": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:stopApplication.sync",
    "Parameters": {
        "ApplicationId": "yourApplicationId"
    },
    "End": true
}
```

Eliminazione di un'applicazione

Il seguente esempio di stato Task elimina un'applicazione utilizzando l'API di integrazione del servizio DeleteApplication.sync.

```
"Delete_Application": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:deleteApplication.sync",
    "Parameters": {
        "ApplicationId": "yourApplicationId"
    },
    "End": true
}
```

Avviare un processo in un'applicazione

Il seguente esempio di stato Task avvia un processo in un'applicazione utilizzando l'API di integrazione del servizio 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
}
```

Annullare un lavoro in un'applicazione

Il seguente esempio di stato Task annulla un processo in un'applicazione utilizzando l'API di integrazione del servizio cancelJobRun.sync.

```
"Cancel_Job": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:cancelJobRun.sync",
    "Parameters": {
        "ApplicationId.$": "$.ApplicationId",
        "JobRunId.$": "$.JobRunId"
    },
    "End": true
}
```

Chiama EventBridge con Step Functions

Step Functions può controllare determinati AWS servizi direttamente da Amazon States Language (ASL). Per ulteriori informazioni, consultare Uso di altri servizi e Passa i parametri a un'API di servizio.

> In che modo l' EventBridge integrazione ottimizzata è diversa dall'integrazione EventBridge AWS SDK
>
> - L'ARN di esecuzione e l'ARN della macchina a stati vengono aggiunti automaticamente al campo di ciascuno. `Resources` `PutEventsRequestEntry`
> - Se la risposta di `PutEvents` contiene un valore diverso da zero, `FailedEntryCount` lo `Task` stato fallisce con l'errore. `EventBridge.FailedEntry`

Per informazioni su come configurare le IAM autorizzazioni quando vengono utilizzate Step Functions con altri AWS servizi, vedere. Politiche IAM per servizi integrati

Step Functions fornisce un'API di integrazione dei servizi per l'integrazione con Amazon EventBridge. Ciò consente di creare applicazioni basate sugli eventi inviando eventi personalizzati direttamente dai flussi di lavoro di Step Functions.

Per utilizzare l'`PutEvents`API, dovrai creare una EventBridge regola nel tuo account che corrisponda allo schema specifico degli eventi che invierai. È ad esempio possibile:

- Crea una funzione Lambda nel tuo account che riceva e stampa un evento che corrisponde a una EventBridge regola.
- Crea una EventBridge regola nel tuo account sul bus eventi predefinito che corrisponda a uno schema di eventi specifico e abbia come destinazione la funzione Lambda.

Per ulteriori informazioni, consultare:

- Aggiungere EventBridge eventi Amazon PutEvents nella Guida per l' EventBridge utente.
- Attendere un callback con il token dell'attivitàin Modelli di integrazione dei servizi.

> **Note**
>
> Esiste una quota per la dimensione massima dei dati di input o risultato per un'attività in Step Functions. Ciò limita a 256 KB di dati come stringa con codifica UTF-8 quando si inviano o si ricevono dati da un altro servizio. Per informazioni, consulta Quote relative alle esecuzioni di macchine a stati.

API supportata EventBridge

EventBridge L'API e la sintassi supportate includono:

- `PutEvents`
  - Sintassi della richiesta
  - Parametro supportato:
    - `Entries`
  - Sintassi della risposta

Quanto segue include un messaggio `Task` che invia un evento personalizzato:

```
{
  "Type": "Task",
  "Resource": "arn:aws:states:::events:putEvents",
  "Parameters": {
    "Entries": [
      {
        "Detail": {
          "Message": "MyMessage"
        },
        "DetailType": "MyDetailType",
        "EventBusName": "MyEventBus",
        "Source": "my.source"
      }
    ]
  },
  "End": true
}
```

Gestione degli errori

L'`PutEvents`API accetta una matrice di voci come input, quindi restituisce una matrice di voci di risultati. Se l'`PutEvents`azione ha avuto esito positivo, `PutEvents` restituirà una risposta HTTP 200, anche se una o più voci non sono riuscite. `PutEvents`restituisce il numero di immissioni non riuscite nel `FailedEntryCount` campo.

Step Functions verifica se `FailedEntryCount` è maggiore di zero. Se è maggiore di zero, Step Functions non riporta lo stato con l'errore`EventBridge.FailedEntry`. Ciò consente di utilizzare la gestione degli errori integrata di Step Functions sugli stati delle attività per rilevare o riprovare in caso di immissioni non riuscite, anziché dover utilizzare uno stato aggiuntivo per analizzare i risultati `FailedEntryCount` della risposta.

> **Note**
>
> Se hai implementato l'idempotenza e puoi riprovare in sicurezza su tutte le voci, puoi utilizzare la logica di riprova di Step Functions. Step Functions non rimuove le voci riuscite dall'array `PutEvents` di input prima di riprovare. Riprova invece con l'array di voci originale.

Gestisci i AWS Glue lavori con Step Functions

Step Functions può controllare determinati AWS servizi direttamente da Amazon States Language (ASL). Per ulteriori informazioni, consultare Uso di altri servizi e Passa i parametri a un'API di servizio.

> **In che modo l' AWS Glue integrazione ottimizzata è diversa dall'integrazione AWS GlueAWS SDK**
>
> - Il modello di Esegui un processo (.sync) integrazione è disponibile.
> - Il `JobName` campo viene estratto dalla richiesta e inserito nella risposta, che normalmente contiene `JobRunID` solo.

AWS Glue API supportata:

- `StartJobRun`

> **I parametri in Step Functions sono espressi in PascalCase**
>
> Anche se l'API del servizio nativo è in CamelCase, ad esempio l'`startSyncExecution`azione API, si specificano parametri PascalCase in, ad esempio:. `StateMachineArn`

Quanto segue include uno `Task` stato che avvia un AWS Glue processo.

```
"Glue StartJobRun": {
      "Type": "Task",
      "Resource": "arn:aws:states:::glue:startJobRun.sync",
      "Parameters": {
        "JobName": "GlueJob-JTrRO5l98qMG"
      },
      "Next": "ValidateOutput"
    },
```

Per informazioni su come configurare IAM le autorizzazioni quando vengono utilizzate Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Gestisci i AWS Glue DataBrew lavori con Step Functions

Step Functions può controllare determinati AWS servizi direttamente da Amazon States Language (ASL). Per ulteriori informazioni, consultare Uso di altri servizi e Passa i parametri a un'API di servizio.

Puoi utilizzare l' DataBrew integrazione per aggiungere fasi di pulizia e normalizzazione dei dati nei flussi di lavoro di analisi e apprendimento automatico.

API supportate DataBrew :

- `StartJobRun`

Quanto segue include `Task` uno stato che avvia un processo di richiesta-risposta DataBrew.

```
"DataBrew StartJobRun": {
            "Type": "Task",
            "Resource": "arn:aws:states:::databrew:startJobRun",
            "Parameters": {
```

```
               "Name": "sample-proj-job-1"
            },
            "Next": "NEXT_STATE"
          },
```

Quanto segue include `Task` uno stato che avvia un processo di sincronizzazione DataBrew .

```
"DataBrew StartJobRun": {
           "Type": "Task",
           "Resource": "arn:aws:states:::databrew:startJobRun.sync",
           "Parameters": {
              "Name": "sample-proj-job-1"
           },
           "Next": "NEXT_STATE"
          },
```

Per informazioni su come configurare IAM le autorizzazioni quando vengono utilizzate Step Functions con altri AWS servizi, consultaPolitiche IAM per servizi integrati.

Richiama Lambda con Step Functions

Step Functions può controllare determinati AWS servizi direttamente da Amazon States Language (ASL). Per ulteriori informazioni, consultare Uso di altri servizi e Passa i parametri a un'API di servizio.

> In che modo l'integrazione Optimized Lambda è diversa dall'integrazione Lambda SDK AWS
>
> - Il `Payload` campo della risposta viene analizzato da Json in escape a Json.
> - Se la risposta contiene il campo `FunctionError` o viene sollevata un'eccezione all'interno della funzione Lambda, l'attività ha esito negativo.

Per ulteriori informazioni sulla gestione di input, output e risultati dello stato, consulta Elaborazione di input e output in Step Functions.

AWS Lambda API supportate:

- `Invoke`

- Sintassi della richiesta
- Parametri supportati
  - `ClientContext`
  - `FunctionName`
  - `InvocationType`
  - `Qualifier`
  - `Payload`
- Sintassi della risposta

> **I parametri in Step Functions sono espressi in PascalCase**
>
> Anche se l'API del servizio nativo è in CamelCase, ad esempio l'`startSyncExecution`azione API, si specificano parametri PascalCase in, ad esempio:. `StateMachineArn`

Quanto segue include uno `Task` stato che richiama una funzione 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
      }
   }
}
```

L'esempio seguente include uno stato `Task` che implementa il modello di integrazione del servizio di 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
      }
   }
}
```

Quando richiami una funzione Lambda, l'esecuzione attenderà il completamento della funzione. Se si richiama la funzione Lambda con un'attività di callback, il timeout del battito cardiaco inizia a contare solo dopo che la funzione Lambda non ha completato l'esecuzione e restituito un risultato. Finché la funzione Lambda è in esecuzione, il timeout del battito cardiaco non viene applicato.

È anche possibile chiamare Lambda in modo asincrono utilizzando il `InvocationType` parametro, come illustrato nell'esempio seguente:

> **Note**
>
> Per le chiamate asincrone delle funzioni Lambda, il periodo di timeout dell'heartbeat inizia immediatamente.

```
{

  "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
    }
  }
}
```

Quando viene restituito il `Task` risultato, l'output della funzione viene annidato all'interno di un dizionario di metadati. Per esempio:

```
{

   "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
}
```

In alternativa, puoi richiamare una funzione Lambda specificando l'ARN di una funzione direttamente nel campo «Risorsa». Quando si richiama una funzione Lambda in questo modo, non è possibile `.waitForTaskToken` specificare e il risultato dell'attività contiene solo l'output della funzione.

```
{
   "StartAt":"CallFunction",
   "States":{
      "CallFunction": {
```

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

Puoi richiamare una versione o un alias specifico della funzione Lambda specificando tali opzioni nell'ARN nel campo. `Resource` Vedi quanto segue nella documentazione di Lambda:

- Funzione Versioni multiple di AWS Lambda
- AWS Lambda alias

Per informazioni su come configurare le IAM autorizzazioni quando vengono utilizzate Step Functions con altri AWS servizi, vedere. Politiche IAM per servizi integrati

Gestisci AWS Elemental MediaConvert con Step Functions

> **Sperimenta con Step Functions e MediaConvert**
>
> Scopri come utilizzare l'integrazione MediaConvert ottimizzata in un flusso di lavoro che rileva e rimuove le barre di colore SMTPE di lunghezza sconosciuta dall'inizio di un video clip. Leggi il post sul blog del 12 aprile 2024: Flussi di lavoro a basso codice con AWS Elemental MediaConvert

Step Functions può controllare determinati AWS servizi direttamente da Amazon States Language (ASL). Per ulteriori informazioni, consultare Uso di altri servizi e Passa i parametri a un'API di servizio.

> **In che modo l'integrazione ottimizzata è diversa dall'integrazione AWS SDK standard**
>
> - Il modello di Esegui un processo (.sync) integrazione è disponibile.
> - Nessuna ottimizzazione Richiesta e risposta o modello di Attendere un callback con il token dell'attività integrazione.

MediaConvert API supportate:

- `CreateJob`
  - Sintassi della richiesta
  - Parametri supportati:
    - `Role` (Obbligatorio)
    - `Settings` (Obbligatorio)
    - `CreateJobRequest` (facoltativo).
  - Sintassi della risposta: vedi schema CreateJobResponse

Quanto segue include uno `Task` stato che invia un MediaConvert lavoro e attende che venga completato.

```
{
    "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
        }
    }
}
```

Per informazioni su come configurare le IAM autorizzazioni quando si utilizza Step Functions con MediaConvert, vedere. Politiche IAM per AWS Elemental MediaConvert

I parametri in Step Functions sono espressi in PascalCase

Anche se l'API del servizio nativo è in CamelCase, ad esempio l'`startSyncExecution`azione API, si specificano parametri PascalCase in, ad esempio:. `StateMachineArn`

Gestisci SageMaker con Step Functions

Step Functions può controllare determinati AWS servizi direttamente da Amazon States Language (ASL). Per ulteriori informazioni, consultare Uso di altri servizi e Passa i parametri a un'API di servizio.

In che modo l' SageMaker integrazione ottimizzata è diversa dall'integrazione SageMaker AWS SDK

- Il modello di Esegui un processo (.sync) integrazione è supportato.
- Non ci sono ottimizzazioni per il modello di Richiesta e risposta integrazione.
- Il modello di Attendere un callback con il token dell'attività integrazione non è supportato.

SageMaker API e sintassi supportate:

- `CreateEndpoint`
  - Sintassi della richiesta
  - Parametri supportati:
    - `EndpointConfigName`
    - `EndpointName`
    - `Tags`
  - Sintassi della risposta
- `CreateEndpointConfig`
  - Sintassi della richiesta
  - Parametri supportati:
    - `EndpointConfigName`
    - `KmsKeyId`

    - ProductionVariants
    - Tags
  - Sintassi della risposta
- CreateHyperParameterTuningJob

> **Note**
>
> Questa azione API supporta il modello di .sync integrazione.

  - Sintassi della richiesta
  - Parametri supportati:
    - HyperParameterTuningJobConfig
    - HyperParameterTuningJobName
    - Tags
    - TrainingJobDefinition
    - WarmStartConfig
  - Sintassi della risposta
- CreateLabelingJob

> **Note**
>
> Questa azione API supporta il modello di .sync integrazione.

  - Sintassi della richiesta
  - Parametri supportati:
    - HumanTaskConfig
    - InputConfig
    - LabelAttributeName
    - LabelCategoryConfigS3Uri
    - LabelingJobAlgorithmsConfig
    - LabelingJobName

    - OutputConfig
    - RoleArn
    - StoppingConditions
    - Tags
  - Sintassi della risposta
- CreateModel
  - Sintassi della richiesta
  - Parametri supportati:
    - Containers
    - EnableNetworkIsolation
    - ExecutionRoleArn
    - ModelName
    - PrimaryContainer
    - Tags
    - VpcConfig
- CreateProcessingJob

> **Note**
>
> Questa azione API supporta il modello di `.sync` integrazione.

  - Sintassi della richiesta
  - Parametri supportati:
    - AppSpecification
    - Environment
    - ExperimentConfig
    - NetworkConfig
    - ProcessingInputs
    - ProcessingJobName
    - ProcessingOutputConfig
    - ProcessingResources

    - RoleArn
    - StoppingCondition
    - Tags
  - Sintassi della risposta
- CreateTrainingJob

> **Note**
>
> Questa azione API supporta il modello di .sync integrazione.

  - Sintassi della richiesta
  - Parametri supportati:
    - AlgorithmSpecification
    - HyperParameters
    - InputDataConfig
    - OutputDataConfig
    - ResourceConfig
    - RoleArn
    - StoppingCondition
    - Tags
    - TrainingJobName
    - VpcConfig
  - Sintassi della risposta
- CreateTransformJob

> **Note**
>
> Questa azione API supporta il modello di .sync integrazione.

> **Note**
>
> AWS Step Functions non creerà automaticamente una politica per`CreateTransformJob`. Devi allegare una policy inline al ruolo creato. Per ulteriori informazioni, consulta questo esempio di politica IAM:`CreateTrainingJob`.

- Sintassi della richiesta
- Parametri supportati:
  - `BatchStrategy`
  - `Environment`
  - `MaxConcurrentTransforms`
  - `MaxPayloadInMB`
  - `ModelName`
  - `Tags`
  - `TransformInput`
  - `TransformJobName`
  - `TransformOutput`
  - `TransformResources`
- Sintassi della risposta
- `UpdateEndpoint`
  - Sintassi della richiesta
  - Parametri supportati:
    - `EndpointConfigName`
    - `EndpointName`
  - Sintassi della risposta

SageMaker Esempio di lavoro Transform

Quanto segue include uno `Task` stato che crea un processo di SageMaker trasformazione di Amazon, specificando la posizione `DataSource` Amazon S3 per e. `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 Esempio di lavoro in Training

Quanto segue include uno `Task` stato che crea un lavoro di SageMaker formazione su 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 Esempio di lavoro in Labeling

Quanto segue include uno `Task` stato che crea un processo di SageMaker etichettatura 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 Esempio di Processing Job

Quanto segue include uno `Task` stato che crea un processo di SageMaker elaborazione 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."
    }
  }
}
```

Per informazioni su come configurare IAM le autorizzazioni quando vengono utilizzate Step Functions con altri AWS servizi, consultaPolitiche IAM per servizi integrati.

Chiama Amazon SNS con Step Functions

Step Functions può controllare determinati AWS servizi direttamente da Amazon States Language (ASL). Per ulteriori informazioni, consultare Uso di altri servizi e Passa i parametri a un'API di servizio.

> In che modo l'integrazione ottimizzata di Amazon SNS è diversa dall'integrazione con Amazon AWS SNS SDK
>
> Non ci sono ottimizzazioni per i modelli di integrazione. Richiesta e risposta Attendere un callback con il token dell'attività

API Amazon SNS supportate:

> Note
>
> Esiste una quota per la dimensione massima dei dati di input o risultato per un'attività in Step Functions. Ciò limita a 256 KB di dati come stringa con codifica UTF-8 quando si inviano o si ricevono dati da un altro servizio. Per informazioni, consulta Quote relative alle esecuzioni di macchine a stati.

- `Publish`
  - Sintassi della richiesta
  - Parametri supportati
    - `Message`
    - `MessageAttributes`
    - `MessageStructure`
    - `PhoneNumber`
    - `Subject`
    - `TargetArn`
    - `TopicArn`

- Sintassi della risposta

> I parametri in sono espressi in Step Functions PascalCase
>
> Anche se l'API del servizio nativo è in CamelCase, ad esempio l'`startSyncExecution`azione API, si specificano parametri PascalCase in, ad esempio:. `StateMachineArn`

Quanto segue include uno `Task` stato che pubblica su un argomento di 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
    }
  }
}
```

Passaggio di valori dinamici. Puoi modificare l'esempio precedente per passare dinamicamente un attributo da questo payload JSON:

```
{
"input": {
  "message": "Hello world"
},
"SNSDetails": {
  "attribute1": "some value",
  "attribute2": "some other value",
}
}
```

Aggiungi il al campo: `.$` `StringValue`

```
"MessageAttributes": {
  "my_attribute_no_1": {
      "DataType": "String",
      "StringValue.$": "$.SNSDetails.attribute1"
  },
  "my_attribute_no_2": {
      "DataType": "String",
      "StringValue.$": "$.SNSDetails.attribute2"
  }
```

Quanto segue include uno `Task` stato che pubblica su un argomento di Amazon SNS e quindi attende la restituzione del token dell'attività. Per informazioni, consulta Attendere un callback con il token dell'attività.

```
{
   "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
      }
   }
```

```
}
```

Per informazioni su come configurare le IAM autorizzazioni quando vengono utilizzate Step Functions con altri AWS servizi, consulta. Politiche IAM per servizi integrati

Chiama Amazon SQS con Step Functions

Step Functions può controllare determinati AWS servizi direttamente da Amazon States Language (ASL). Per ulteriori informazioni, consultare Uso di altri servizi e Passa i parametri a un'API di servizio.

> In che modo l'integrazione ottimizzata di Amazon SQS è diversa dall'integrazione con Amazon SQS SDK AWS
>
> Non ci sono ottimizzazioni per i modelli di integrazione. Richiesta e risposta Attendere un callback con il token dell'attività

API Amazon SQS supportate:

> **Note**
>
> Esiste una quota per la dimensione massima dei dati di input o risultato per un'attività in Step Functions. Ciò limita a 256 KB di dati come stringa con codifica UTF-8 quando si inviano o si ricevono dati da un altro servizio. Per informazioni, consulta Quote relative alle esecuzioni di macchine a stati.

- `SendMessage`

  Parametri supportati:

  - `DelaySeconds`
  - `MessageAttribute`
  - `MessageBody`
  - `MessageDeduplicationId`
  - `MessageGroupId`
  - `QueueUrl`

- Response syntax

> I parametri in sono espressi in Step Functions PascalCase
>
> Anche se l'API del servizio nativo è in CamelCase, ad esempio l'`startSyncExecution`azione API, si specificano parametri PascalCase in, ad esempio:. `StateMachineArn`

Quanto segue include uno `Task` stato che invia un messaggio 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
    }
  }
}
```

Quanto segue include uno `Task` stato che pubblica su una coda Amazon SQS e quindi attende la restituzione del token dell'attività. Per informazioni, consulta Attendere un callback con il token dell'attività.

```
{
   "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
      }
   }
}
```

Per ulteriori informazioni sulla ricezione di messaggi in Amazon SQS, consulta Receive and Delete Your Message nella Amazon Simple Queue Service Developer Guide.

Per informazioni su come configurare le IAM autorizzazioni quando vengono utilizzate Step Functions con altri AWS servizi, consulta. Politiche IAM per servizi integrati

Gestisci AWS Step Functions le esecuzioni come servizio integrato

Step Functions si integra con la propria API come integrazione di servizi. Ciò consente a Step Functions di avviare una nuova esecuzione di una macchina a stati direttamente dallo stato dell'attività di un'esecuzione in esecuzione. Durante la creazione di nuovi flussi di lavoro, utilizza le esecuzioni di flusso di lavoro nidificate per ridurre la complessità dei flussi di lavoro principali e riutilizzare i processi comuni.

In che modo l'integrazione Optimized Step Functions è diversa dall'integrazione Step Functions AWS SDK

- Il modello di Esegui un processo (.sync) integrazione è disponibile.

Nota che non ci sono ottimizzazioni per i modelli Richiesta e risposta di Attendere un callback con il token dell'attività integrazione.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Avvio delle esecuzioni da un'attività
- Uso di altri servizi
- Passa i parametri a un'API di servizio

API e sintassi Step Functions supportate:

- `StartExecution`
  - Sintassi della richiesta
  - Parametri supportati
    - `Input`
    - `Name`
    - `StateMachineArn`
  - Sintassi della risposta

Di seguito è incluso uno stato `Task` che avvia l'esecuzione di un'altra macchina a stati e ne attende il completamento.

```
{
   "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
}
```

Di seguito è incluso uno stato `Task` che avvia l'esecuzione di un'altra macchina a stati.

```
{
   "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
}
```

L'esempio seguente include uno stato `Task` che implementa il modello di integrazione del servizio di 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
}
```

Per associare un'esecuzione di flusso di lavoro nidificata all'esecuzione padre che l'ha avviata, trasferisce un parametro appositamente denominato che include l'ID di esecuzione estratto dall'oggetto contesto. Quando si avvia un'esecuzione nidificata, utilizza un parametro denominato `AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID`. Passa l'ID di esecuzione aggiungendo `.$` al nome del parametro e facendo riferimento all'ID nell'oggetto contesto con `$$.Execution.Id`. Per ulteriori informazioni, consulta Accesso all'oggetto contestuale.

```
{
   "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
}
```

Le macchine a stato nidificato restituiscono quanto segue:

Risorsa	Output
startExecution.sync	Stringa
startExecution.sync:2	JSON

Entrambi attenderanno il completamento della macchina a stati nidificati, ma restituiscono formati `Output` diversi. Ad esempio, se crei una funzione Lambda che restituisce l'oggetto`{ "MyKey": "MyValue" }`, otterrai le seguenti risposte:

Per startExecution.sync:

```
{
   <other fields>
   "Output": "{ \"MyKey\": \"MyValue\" }"
}
```

Per startExecution.sync:2:

```
{
   <other fields>
   "Output": {
      "MyKey": "MyValue"
   }
}
```

Configurazione delle autorizzazioni IAM per macchine a stati annidate

Una macchina a stati principale determina se una macchina a stati figlio ha completato l'esecuzione utilizzando polling ed eventi. Il polling richiede l'autorizzazione `states:DescribeExecution` mentre gli eventi inviati EventBridge a Step Functions richiedono le autorizzazioni per `events:PutTargetsevents:PutRule`, e. `events:DescribeRule` Se nel tuo ruolo IAM mancano queste autorizzazioni, potrebbe verificarsi un ritardo prima che una macchina a stati principale venga a conoscenza del completamento dell'esecuzione della macchina a stati secondari.

Per una macchina a stati che richiede `StartExecution` l'esecuzione di un singolo flusso di lavoro annidato, utilizza una policy IAM che limiti le autorizzazioni a quella macchina a stati.

Per ulteriori informazioni, consulta Autorizzazioni IAM per Step Functions.

Chiama API di terze parti

Un Task HTTP è un tipo di Attività stato che ti consente di richiamare qualsiasi API pubblica di terze parti, come Salesforce e Stripe, nei tuoi flussi di lavoro. Per chiamare un'API di terze parti, utilizza lo stato Task con la risorsa. `arn:aws:states:::http:invoke` Quindi, fornisci i dettagli di configurazione dell'endpoint dell'API, come l'URL dell'API, il metodo che desideri utilizzare e i dettagli di autenticazione.

Se utilizzate Workflow Studio per creare una macchina a stati che contiene un task HTTP, Workflow Studio genera automaticamente un ruolo di esecuzione con IAM politiche per il task HTTP. Per ulteriori informazioni, consulta Ruolo per il test delle attività HTTP in Workflow Studio.

Argomenti

- Definizione del task HTTP
- Campi HTTP Task
- Autenticazione per un task HTTP
- Unione dei dati di EventBridge connessione e di definizione delle attività HTTP
- Applicazione della codifica URL sul corpo della richiesta
- Autorizzazioni IAM per eseguire un task HTTP
- Esempio di attività HTTP
- Test di un task HTTP

- Risposte HTTP Task non supportate

Definizione del task HTTP

La definizione ASL rappresenta un task HTTP con `http:invoke` risorsa. La seguente definizione di HTTP Task richiama un'API Stripe che restituisce un elenco di tutti i clienti.

```
"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
}
```

Campi HTTP Task

Un task HTTP include i seguenti campi nella sua definizione.

`Resource` (Obbligatorio)

Per specificare un tipo di attività, inserisci il relativo ARN nel `Resource` campo. Per un task HTTP, si specifica il `Resource` campo come segue.

```
"Resource": "arn:aws:states:::http:invoke"
```

`Parameters` (Obbligatorio)

Contiene i `ConnectionArn` campi `ApiEndpoint``Method`, e che forniscono informazioni sull'API di terze parti che desideri chiamare. `Parameters`contiene anche campi opzionali, come `Headers` e`QueryParameters`.

È possibile specificare una combinazione di JSON statico e JsonPathsintassi come `Parameters` nel `Parameters` campo. Per ulteriori informazioni, consulta Passa i parametri a un'API di servizio.

`ApiEndpoint`(Obbligatorio)

Speciifica l'URL dell'API di terze parti che desideri chiamare. Per aggiungere parametri di query all'URL, utilizza il campo. `QueryParameters` L'esempio seguente mostra come chiamare un'API Stripe per recuperare l'elenco di tutti i clienti.

```
"ApiEndpoint":"https://api.stripe.com/v1/customers"
```

Puoi anche specificare un percorso di riferimento utilizzando la JsonPathsintassi per selezionare il nodo JSON che contiene l'URL dell'API di terze parti. Ad esempio, supponiamo che tu voglia chiamare una delle API di Stripe utilizzando un ID cliente specifico. Immagina di aver fornito il seguente input di stato.

```
{
    "customer_id": "1234567890",
    "name": "John Doe"
}
```

Per recuperare i dettagli di questo ID cliente utilizzando un'API Stripe, specifica `ApiEndpoint` come mostrato nell'esempio seguente. Questo esempio utilizza una funzione intrinseca e un percorso di riferimento.

```
"ApiEndpoint.$":"States.Format('https://api.stripe.com/v1/customers/{}',
 $.customer_id)"
```

In fase di esecuzione, Step Functions risolve il valore di `ApiEndpoint` come segue.

```
https://api.stripe.com/v1/customers/1234567890
```

`Method`(Obbligatorio)

Speciifica il metodo HTTP che desideri utilizzare per chiamare un'API di terze parti. È possibile specificare uno di questi metodi nel task HTTP: GET, POST, PUT, DELETE, PATCH, OPTIONS o HEAD.

Ad esempio, per utilizzare il metodo GET, specificare il `Method` campo come segue.

```
"Method": "GET"
```

È inoltre possibile utilizzare un percorso di riferimento per specificare il metodo in fase di esecuzione. Ad esempio, **`"Method.$": "$.myHTTPMethod"`**.

`Authentication`(Obbligatorio)

Contiene il `ConnectionArn` campo che specifica come autenticare una chiamata API di terze parti. Step Functionssupporta l'autenticazione per una determinata area `ApiEndpoint` utilizzando la risorsa di connessione di. Amazon EventBridge

`ConnectionArn`(Richiesto)

Speciifica l'ARN della EventBridge connessione.

Un task HTTP richiede una EventBridge connessione, che gestisce in modo sicuro le credenziali di autenticazione di un provider di API. Una connessione specifica il tipo di autorizzazione e le credenziali da utilizzare per autorizzare un'API di terze parti. L'utilizzo di una connessione consente di evitare l'inserimento di segreti codificati, come le chiavi API, nella definizione della macchina a stati. In una connessione, potete anche specificare `HeadersQueryParameters`, e parametri. `RequestBody`

Quando si crea una EventBridge connessione, si forniscono i dettagli di autenticazione. Per ulteriori informazioni su come funziona l'autenticazione per un task HTTP, consultaAutenticazione per un task HTTP.

L'esempio seguente mostra come specificare il `Authentication` campo nella definizione del task HTTP.

```
"Authentication": {
  "ConnectionArn": "arn:aws:events:us-east-2:123456789012:connection/
Stripe/81210c42-8af1-456b-9c4a-6ff02fc664ac"
}
```

`Headers` (facoltativo).

Fornisce contesto e metadati aggiuntivi all'endpoint dell'API. È possibile specificare le intestazioni come stringa o array JSON.

È possibile specificare le intestazioni nella EventBridge connessione e nel `Headers` campo in un task HTTP. Ti consigliamo di non includere nel `Headers` campo i dettagli di autenticazione dei tuoi provider di API. Ti consigliamo di includere questi dettagli nella tua EventBridge connessione.

Step Functionsaggiunge le intestazioni specificate nella EventBridge connessione alle intestazioni specificate nella definizione del task HTTP. Se nella definizione e nella connessione sono presenti le stesse chiavi di intestazione, Step Functions utilizza i valori corrispondenti specificati nella EventBridge connessione per tali intestazioni. Per ulteriori informazioni su come Step Functions esegue l'unione dei dati, vedere. Unione dei dati di EventBridge connessione e di definizione delle attività HTTP

L'esempio seguente specifica un'intestazione che verrà inclusa in una chiamata API di terze parti:. `content-type`

```
"Headers": {
  "content-type": "application/json"
}
```

È inoltre possibile utilizzare un percorso di riferimento per specificare le intestazioni in fase di esecuzione. Ad esempio, **`"Headers.$": "$.myHTTPHeaders"`**.

Step Functionsimposta le `Host` intestazioni `User-Agent``Range`, e. Step Functionsimposta il valore dell'`Host`intestazione in base all'API che stai chiamando. Di seguito è riportato un esempio di queste intestazioni.

```
User-Agent: Amazon|StepFunctions|HttpInvoke|us-east-1,
Range: bytes=0-262144,
Host: api.stripe.com
```

Non è possibile utilizzare le seguenti intestazioni nella definizione del task HTTP.
Se si utilizzano queste intestazioni, il task HTTP ha esito negativo e viene restituito l'`States.Runtime`errore.

- A-IM
- Accept-Charset
- Accept-Datetime
- Accept-Encoding
- Cache-Control
- Connessione
- Content-Encoding
- Content-MD5

- Data
- Expect
- Forwarded
- Da
- Host
- sImpostazioni HTTP2
- If-Match
- If-Modified-Since
- If-None-Match
- If-Range
- If-Unmodified-Since
- Max-Forwards
- Origin
- Pragma
- Proxy-Authorization
- Referente
- Server
- TE
- Trailer
- Transfer-Encoding
- Upgrade
- Via
- Attenzione
- x-forwarded-*
- x-amz-*
- x-amzn-*

`QueryParameters` (facoltativo).

Inserisce coppie chiave-valore alla fine di un URL API. È possibile specificare i parametri di query come stringa, array JSON o oggetto JSON. Step Functionsl'URL codifica automaticamente i parametri di interrogazione quando richiama un'API di terze parti.

Ad esempio, supponiamo di voler chiamare l'API Stripe per cercare clienti che effettuano transazioni in dollari USA (USD). Immagina di aver fornito quanto segue `QueryParameters` come input di stato.

```
"QueryParameters": {
  "currency": "usd"
}
```

In fase di esecuzione, Step Functions aggiunge `QueryParameters` all'URL dell'API come segue.

```
https://api.stripe.com/v1/customers/search?currency=usd
```

È inoltre possibile utilizzare un percorso di riferimento per specificare i parametri della query in fase di esecuzione. Ad esempio, **`"QueryParameters.$": "$.myQueryParameters"`**.

Se hai specificato i parametri di query nella EventBridge connessione, Step Functions aggiunge questi parametri di query ai parametri di query specificati nella definizione del task HTTP. Se nella definizione e nella connessione sono presenti gli stessi parametri di query, Step Functions utilizza i valori corrispondenti specificati nella EventBridge connessione per tali intestazioni. Per ulteriori informazioni su come Step Functions esegue l'unione dei dati, vedere. Unione dei dati di EventBridge connessione e di definizione delle attività HTTP

`Transform` (facoltativo).

Contiene i `RequestEncodingOptions` campi `RequestBodyEncoding` e. Per impostazione predefinita, Step Functions invia il corpo della richiesta come dati JSON a un endpoint API.

Se il tuo provider di API accetta i corpi delle `form-urlencoded` richieste, utilizza il `Transform` campo per specificare la codifica URL per i corpi della richiesta. È inoltre necessario specificare l'intestazione come. `content-type` `application/x-www-form-urlencoded` Step Functionsquindi codifica automaticamente l'URL del corpo della richiesta.

`RequestBodyEncoding`

Specifica la codifica URL del corpo della richiesta. È possibile specificare uno di questi valori: o. `NONE` `URL_ENCODED`

- `NONE`— Il corpo della richiesta HTTP sarà il codice JSON serializzato del `RequestBody` campo. Si tratta del valore di default.

- `URL_ENCODED`— Il corpo della richiesta HTTP sarà costituito dai dati del modulo con codifica URL del campo. `RequestBody`

`RequestEncodingOptions`

Determina l'opzione di codifica da utilizzare per gli array nel corpo della richiesta, se impostata su. `RequestBodyEncoding` `URL_ENCODED`

Step Functionssupporta le seguenti opzioni di codifica degli array. Per ulteriori informazioni su queste opzioni e sui relativi esempi, vedereApplicazione della codifica URL sul corpo della richiesta.

- `INDICES`— Codifica gli array utilizzando il valore di indice degli elementi dell'array. Per impostazione predefinita, Step Functions utilizza questa opzione di codifica.
- `REPEAT`— Ripete una chiave per ogni elemento di un array.
- `COMMAS`— Codifica tutti i valori di una chiave come elenco di valori delimitato da virgole.
- `BRACKETS`— Ripete una chiave per ogni elemento di una matrice e aggiunge una parentesi, [], alla chiave per indicare che si tratta di una matrice.

L'esempio seguente imposta la codifica URL per i dati del corpo della richiesta. Specifica inoltre di utilizzare l'opzione di `COMMAS` codifica per gli array nel corpo della richiesta.

```
"Transform": {
  "RequestBodyEncoding": "URL_ENCODED",
  "RequestEncodingOptions": {
    "ArrayFormat": "COMMAS"
  }
}
```

`RequestBody` (facoltativo).

Accetta i dati JSON forniti nell'input dello stato. In`RequestBody`, puoi specificare una combinazione di JSON statico e JsonPathsintassi. Ad esempio, supponiamo di fornire il seguente input di stato:

```
{
    "CardNumber": "1234567890",
    "ExpiryDate": "09/25"
}
```

Per utilizzare questi valori di `CardNumber` e `ExpiryDate` nel corpo della richiesta in fase di esecuzione, puoi specificare i seguenti dati JSON nel corpo della richiesta.

```
"RequestBody": {
  "Card": {
    "Number.$": "$.CardNumber",
    "Expiry.$": "$.ExpiryDate",
    "Name": "John Doe",
    "Address": "123 Any Street, Any Town, USA"
  }
}
```

Se l'API di terze parti che desideri chiamare richiede corpi di `form-urlencoded` richiesta, devi specificare la codifica URL per i dati del corpo della richiesta. Per ulteriori informazioni, consulta Applicazione della codifica URL sul corpo della richiesta.

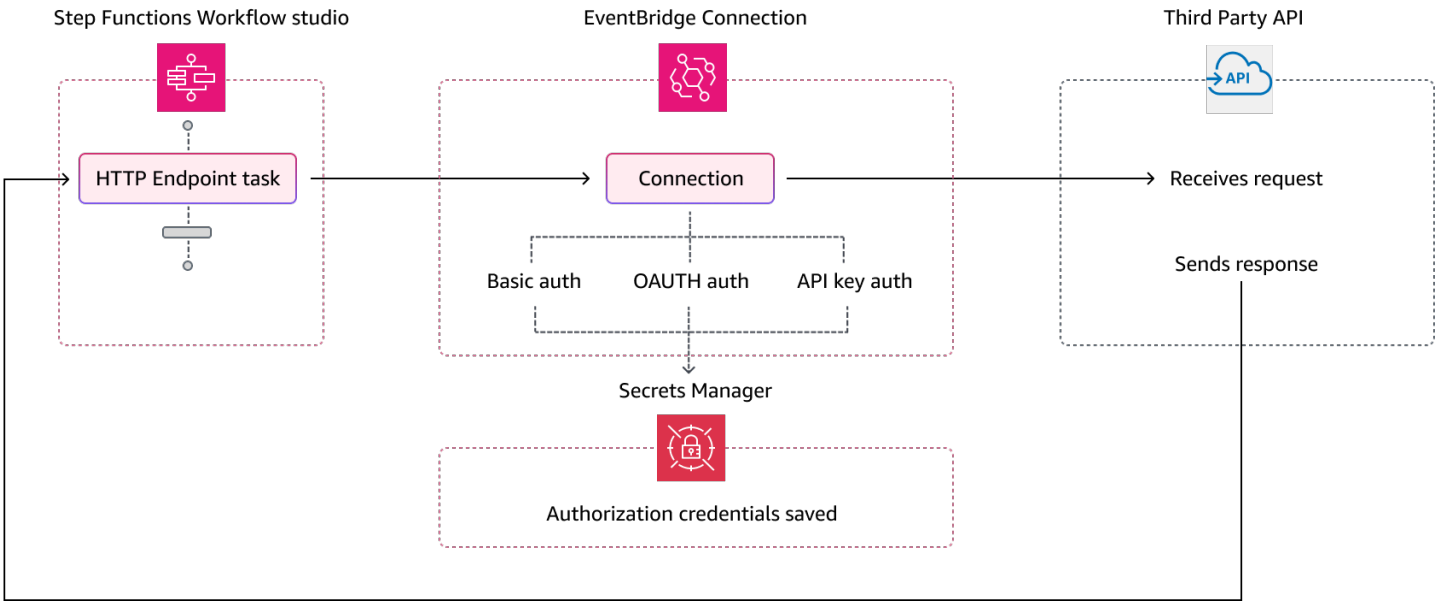
Autenticazione per un task HTTP

Un task HTTP richiede una EventBridge connessione, che gestisce in modo sicuro le credenziali di autenticazione di un provider di API. Una connessione specifica il tipo di autorizzazione e le credenziali da utilizzare per autorizzare un'API di terze parti. L'utilizzo di una connessione consente di evitare l'inserimento di segreti codificati, come le chiavi API, nella definizione della macchina a stati. Una EventBridge connessione supporta gli schemi di autorizzazione Basic, OAuth e API Key.

Quando crei una EventBridge connessione, fornisci i tuoi dati di autenticazione. Puoi anche includere i parametri header, body e query necessari per l'autorizzazione con un'API. È necessario includere l'ARN della connessione in qualsiasi attività HTTP che richiama un'API di terze parti.

Quando si crea una connessione e si aggiungono parametri di autorizzazione, EventBridge crea un indirizzo segreto in AWS Secrets Manager. In questo segreto, EventBridge memorizza i parametri di connessione e autorizzazione in forma crittografata. Per creare o aggiornare correttamente una connessione, è necessario utilizzare un Account AWS utente autorizzato a utilizzare Secrets Manager. Per ulteriori informazioni sulle IAM autorizzazioni necessarie alla macchina a stati per accedere a una EventBridge connessione, vedereAutorizzazioni IAM per eseguire un task HTTP.

L'immagine seguente mostra come Step Functions gestisce l'autenticazione per le chiamate API di terze parti utilizzando una EventBridge connessione.



Unione dei dati di EventBridge connessione e di definizione delle attività HTTP

Quando si richiama un task HTTP, è possibile specificare i dati nella EventBridge connessione e nella definizione del task HTTP. Questi dati includono `Headers` `QueryParameters` e `RequestBody` parametri. Prima di chiamare un'API di terze parti, Step Functions unisce il corpo della richiesta con i parametri del corpo di connessione in tutti i casi, tranne se il corpo della richiesta è una stringa e i parametri del corpo di connessione non sono vuoti. In questo caso, il task HTTP ha esito negativo e restituisce l'`States.Runtime`errore.

Se sono presenti chiavi duplicate specificate nella definizione del task HTTP e nella EventBridge connessione, Step Functions sovrascrive i valori del task HTTP con i valori della connessione.

L'elenco seguente descrive come Step Functions unisce i dati prima di chiamare un'API di terze parti:

- Intestazioni: Step Functions aggiunge le intestazioni specificate nella connessione alle intestazioni nel `Headers` campo del Task HTTP. In caso di conflitto tra le chiavi di intestazione, Step Functions utilizza i valori specificati nella connessione per tali intestazioni. Ad esempio, se hai specificato l'`content-type`intestazione sia nella definizione del task HTTP che nella EventBridge connessione, Step Functions utilizza il valore dell'`content-type`intestazione specificato nella connessione.
- Parametri di query: Step Functions aggiunge tutti i parametri di query specificati nella connessione ai parametri di query nel `QueryParameters` campo del task HTTP. In caso di conflitto tra le chiavi dei parametri di query, Step Functions utilizza i valori specificati nella connessione per

tali parametri di query. Ad esempio, se hai specificato il parametro di `maxItems` query sia nella definizione del task HTTP che nella EventBridge connessione, Step Functions utilizza il valore del parametro di `maxItems` query specificato nella connessione.

- Parametri corpo
  - Step Functionsaggiunge tutti i valori del corpo della richiesta specificati nella connessione al corpo della richiesta nel `RequestBody` campo del task HTTP. In caso di conflitto tra le chiavi del corpo della richiesta, Step Functions utilizza i valori specificati nella connessione per il corpo della richiesta. Ad esempio, supponiamo di aver specificato un `Mode` campo sia nella `RequestBody` definizione del task HTTP che nella EventBridge connessione. Step Functionsutilizza il valore del `Mode` campo specificato nella connessione.
  - Se si specifica il corpo della richiesta come stringa anziché come oggetto JSON e la EventBridge connessione contiene anche il corpo della richiesta, non è Step Functions possibile unire il corpo della richiesta specificato in entrambe queste posizioni. Fallisce l'operazione HTTP con l'`States.Runtime`errore.

  Step Functionsapplica tutte le trasformazioni e serializza il corpo della richiesta dopo aver completato la fusione del corpo della richiesta.

L'esempio seguente imposta i `RequestBody` campi `HeadersQueryParameters`, e sia nel task HTTP che nella connessione. EventBridge

Definizione del task 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"
        }
      }
    }
  }
}
```

connessione EventBridge

```
{
  "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"
        }
      ]
```

```
    }
  }
}
```

In questo esempio, le chiavi duplicate vengono specificate nel task e nella EventBridge connessione HTTP. Pertanto, Step Functions sovrascrive i valori del task HTTP con i valori della connessione. Il seguente frammento di codice mostra la richiesta HTTP Step Functions inviata all'API di terze parti.

```
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"}
```

Applicazione della codifica URL sul corpo della richiesta

Per impostazione predefinita, Step Functions invia il corpo della richiesta come dati JSON a un endpoint API. Se il tuo provider di API di terze parti richiede corpi di `form-urlencoded` richiesta, devi specificare la codifica URL per i corpi della richiesta. Step Functionsquindi codifica automaticamente l'URL del corpo della richiesta in base all'opzione di codifica URL selezionata.

La codifica URL viene specificata utilizzando il campo. `Transform` Questo campo contiene il `RequestBodyEncoding` campo che specifica se si desidera applicare o meno la codifica URL per i corpi della richiesta. Quando specifichi il `RequestBodyEncoding` campo, Step Functions converte il corpo della richiesta JSON in quello della richiesta prima di chiamare l'API di `form-urlencoded` terze parti. È inoltre necessario specificare l'`content-type`intestazione in `application/x-www-form-urlencoded` quanto le API che accettano dati con codifica URL si aspettano l'intestazione. `content-type`

Per codificare gli array nel corpo della richiesta, fornisce le seguenti opzioni di codifica dell'array. Step Functions

- `INDICES`— Ripete una chiave per ogni elemento di un array e aggiunge una parentesi, [], alla chiave per indicare che si tratta di un array. Questa parentesi contiene l'indice dell'elemento

dell'array. L'aggiunta dell'indice consente di specificare l'ordine degli elementi dell'array. Per impostazione predefinita, Step Functions utilizza questa opzione di codifica.

Ad esempio, se il corpo della richiesta contiene il seguente array.

```
{"array": ["a","b","c","d"]}
```

Step Functionscodifica questo array nella seguente stringa.

```
array[0]=a&array[1]=b&array[2]=c&array[3]=d
```

- `REPEAT`— Ripete una chiave per ogni elemento di un array.

  Ad esempio, se il corpo della richiesta contiene il seguente array.

  ```
  {"array": ["a","b","c","d"]}
  ```

  Step Functionscodifica questo array nella seguente stringa.

  ```
  array=a&array=b&array=c&array=d
  ```

- `COMMAS`— Codifica tutti i valori di una chiave come elenco di valori delimitato da virgole.

  Ad esempio, se il corpo della richiesta contiene il seguente array.

  ```
  {"array": ["a","b","c","d"]}
  ```

  Step Functionscodifica questo array nella seguente stringa.

  ```
  array=a,b,c,d
  ```

- `BRACKETS`— Ripete una chiave per ogni elemento di un array e aggiunge una parentesi, [], alla chiave per indicare che si tratta di una matrice.

  Ad esempio, se il corpo della richiesta contiene il seguente array.

  ```
  {"array": ["a","b","c","d"]}
  ```

  Step Functionscodifica questo array nella seguente stringa.

```
array[]=a&array[]=b&array[]=c&array[]=d
```

Autorizzazioni IAM per eseguire un task HTTP

Il ruolo di esecuzione della macchina a stati deve disporre delle `secretsmanager:DescribeSecret` autorizzazioni `states:InvokeHTTPEndpoint` `events:RetrieveConnectionCredentialssecretsmanager:GetSecretValue`,, e affinché un task HTTP possa chiamare un'API di terze parti. Il seguente esempio di policy IAM concede i privilegi minimi richiesti al ruolo di macchina statale per chiamare le API Stripe. Questa policy IAM concede inoltre l'autorizzazione al ruolo della macchina a stati per accedere a una EventBridge connessione specifica, incluso il segreto per questa connessione archiviato in 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/*"
        }
    ]
}
```

Esempio di attività HTTP

La seguente definizione di macchina a stati mostra un task HTTP che include i `RequestBody` parametri `Headers` `QueryParametersTransform`,, e. L'HTTP Task richiama un'API Stripe, https://api.stripe.com/v1/invoices, per generare una fattura. L'HTTP Task specifica anche la codifica URL per il corpo della richiesta utilizzando l'opzione di codifica. `INDICES`

Assicurati di aver creato una connessione. EventBridge L'esempio seguente mostra una connessione creata utilizzando il tipo di autenticazione BASIC.

```
{
    "Type": "BASIC",
    "AuthParameters": {
        "BasicAuthParameters": {
            "Password": "myPassword",
            "Username": "myUsername"
         },
    }
}
```

Ricordati di sostituire il testo in *`corsivo`* con le informazioni specifiche della risorsa.

```
{
  "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
    }
  }
}
```

Per eseguire questa macchina a stati, fornisci l'ID cliente come input, come mostrato nell'esempio seguente:

```
{
    "customer_id": "1234567890"
}
```

L'esempio seguente mostra la richiesta HTTP Step Functions inviata all'API Stripe.

```
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
```

Test di un task HTTP

Puoi utilizzare l'TestStateAPI tramite la console, l'SDK o AWS CLI per testare un task HTTP. La procedura seguente descrive come utilizzare l' TestState API nella Step Functions console. Puoi testare in modo iterativo la richiesta, la risposta e i dettagli di autenticazione dell'API finché l'HTTP Task non funzioni come previsto.

Verifica lo stato di un task HTTP nella console Step Functions

1. Apri la console Step Functions.

2. Scegli Crea macchina a stati per iniziare a creare una macchina a stati o scegli una macchina a stati esistente che contiene un task HTTP.

   Fai riferimento al Passaggio 4 se stai testando l'attività in una macchina a stati esistente.

3. In Workflow Studio, configura visivamente un'attività HTTP. Modalità di progettazione Oppure scegli la modalità Codice per copiare e incollare la definizione della macchina a stati dal tuo ambiente di sviluppo locale.

4. In modalità Progettazione, scegliete Test state nel ✓Inspector pannello di Workflow Studio.

5. Nella finestra di dialogo Test state, effettuate le seguenti operazioni:

   a. Per Ruolo di esecuzione, scegliete un ruolo di esecuzione per testare lo stato. Se non disponi di un ruolo con autorizzazioni sufficienti per un'attività HTTP, consulta Ruolo per il test delle attività HTTP in Workflow Studio per creare un ruolo.

   b. (Facoltativo) Fornisci qualsiasi input JSON necessario allo stato selezionato per il test.

   c. Per il livello di ispezione, mantieni la selezione predefinita di INFO. Questo livello mostra lo stato della chiamata API e lo stato dell'output. Questo è utile per verificare rapidamente la risposta dell'API.

   d. Scegli Avvia test.

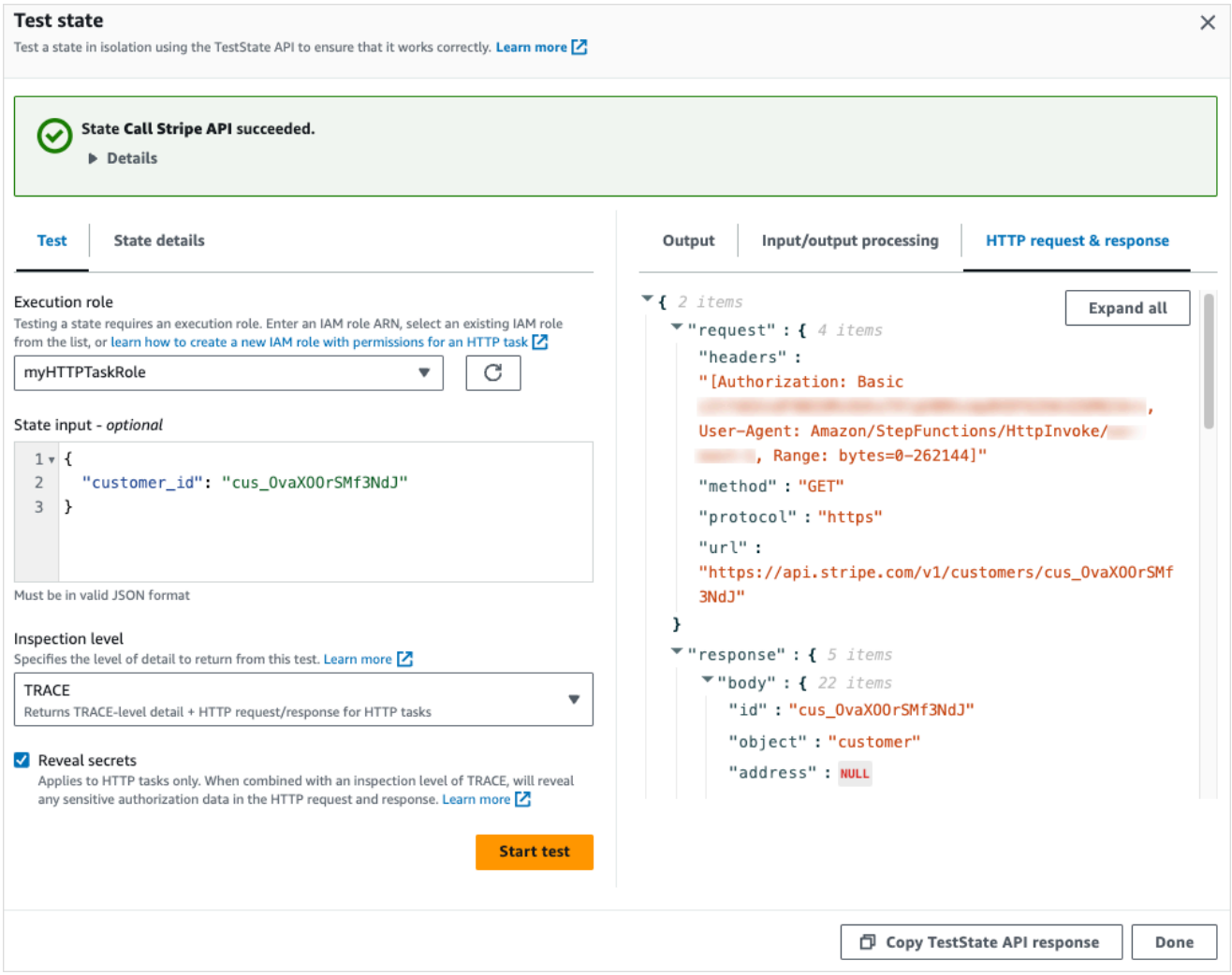
   e. Se il test ha esito positivo, l'output dello stato viene visualizzato sul lato destro della finestra di dialogo Test state. Se il test fallisce, viene visualizzato un errore.

      Nella scheda Dettagli dello stato della finestra di dialogo, puoi vedere la definizione dello stato e un link alla tua EventBridgeconnessione.

   f. Modificate il livello di ispezione in TRACE. Questo livello mostra la richiesta e la risposta HTTP non elaborate ed è utile per verificare intestazioni, parametri di query e altri dettagli specifici dell'API.

   g. Scegli la casella di controllo Rivela segreti. In combinazione con TRACE, questa impostazione consente di visualizzare i dati sensibili inseriti dalla EventBridge connessione, come le chiavi API. L'identità IAM utente che utilizzi per accedere alla console deve disporre dell'autorizzazione per eseguire l'`states:RevealSecrets`azione. Senza questa autorizzazione, all'avvio Step Functions del test viene generato un errore di accesso negato. Per un esempio di IAM policy che imposta l'`states:RevealSecrets`autorizzazione, vediIAMautorizzazioni per l'utilizzo dell'API TestState .

L'immagine seguente mostra un test per un task HTTP che ha esito positivo. Il livello di ispezione per questo stato è impostato su TRACE. La scheda Richiesta e risposta HTTP nell'immagine seguente mostra il risultato della chiamata API di terze parti.



h. Scegli Avvia test.

i. Se il test ha esito positivo, puoi visualizzare i dettagli HTTP nella scheda Richiesta e risposta HTTP.

Risposte HTTP Task non supportate

Un task HTTP ha esito negativo e restituisce l'`States.Runtime`errore se una delle seguenti condizioni è vera per la risposta restituita:

- La risposta contiene un'intestazione del tipo di contenuto di`application/octet-stream`,`image/*`, `video/*` o. `audio/*`
- La risposta non può essere letta come una stringa valida. Ad esempio, dati binari o di immagine.

Modelli di integrazione dei servizi

AWS Step Functions si integra con i servizi direttamente in Amazon States Language. È possibile controllare questi servizi AWS utilizzando tre modelli di integrazione del servizio.

- Chiama un servizio e lascia che Step Functions passi allo stato successivo subito dopo aver ricevuto una risposta HTTP.
- Chiama un servizio e chiedi a Step Functions di attendere il completamento di un lavoro.
- Chiama un servizio con un task token e chiedi a Step Functions di attendere che quel token venga restituito con un payload.

Ognuno di questi modelli di integrazione del servizio è controllato dal modo in cui si crea una URI nel campo `"Resource"` della definizione dell'attività.

Metodi per chiamare un servizio integrato

- Richiesta e risposta
- Esegui un processo (.sync)
- Attendere un callback con il token dell'attività

Per informazioni sulla configurazione AWS Identity and Access Management (IAM) per i servizi integrati, consulta. Politiche IAM per servizi integrati

Richiesta e risposta

Quando si specifica un servizio nella `"Resource"` stringa dello stato dell'attività e si fornisce solo la risorsa, Step Functions attenderà una risposta HTTP e quindi passerà allo stato successivo. Step Functions non aspetterà il completamento di un lavoro.

L'esempio seguente mostra come pubblicare un argomento di 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"
}
```

Questo esempio fa riferimento all'API Publish di Amazon SNS. Il flusso di lavoro avanza allo stato successivo dopo aver richiamato l'API `Publish`.

> **Tip**
>
> Per implementare un flusso di lavoro di esempio che utilizza il modello di integrazione del servizio Request Response sul tuo Account AWS, consulta il Modulo 2 - Request Response of The AWS Step Functions Workshop.

Esegui un processo (.sync)

Per i servizi integrati come AWS Batch Amazon ECS, Step Functions può attendere il completamento di una richiesta prima di passare allo stato successivo. Per far attendere Step Functions, specifica il `"Resource"` campo nella definizione dello stato dell'attività con il `.sync` suffisso aggiunto dopo l'URI della risorsa.

Ad esempio, quando inviate un AWS Batch lavoro, utilizzate il `"Resource"` campo nella definizione della macchina a stati, come illustrato in questo esempio.

```
"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"
}
```

Se la `.sync` parte viene aggiunta alla risorsa Amazon Resource Name (ARN), Step Functions attende il completamento del lavoro. Dopo la chiamata AWS Batch `submitJob`, il flusso di lavoro

si mette in pausa. Una volta completato il lavoro, Step Functions passa allo stato successivo. Per ulteriori informazioni, vedere il progetto AWS Batch di esempio:Gestire un processo batch (AWS Batch,Amazon SNS).

Se un'attività che utilizza questo modello di integrazione del servizio (`.sync`) viene interrotta e Step Functions non è in grado di annullarla, potrebbero essere addebitati costi aggiuntivi dal servizio integrato. Un'attività può essere interrotta se:

- L'esecuzione della macchina a stati viene interrotta.
- Un ramo diverso di uno stato parallelo fallisce con un errore non rilevato.
- Un'iterazione di uno stato della mappa fallisce e viene rilevato un errore non rilevato.

Step Functions farà del suo meglio per annullare l'operazione. Ad esempio, se un'`states:startExecution.sync`attività Step Functions viene interrotta, chiamerà l'azione `StopExecution` API Step Functions. Tuttavia, è possibile che Step Functions non sia in grado di annullare l'operazione. I motivi di ciò includono, a titolo esemplificativo ma non esaustivo:

- Il tuo ruolo di esecuzione IAM non è autorizzato a effettuare la chiamata API corrispondente.
- Si è verificata un'interruzione temporanea del servizio.

Quando si utilizza il modello di integrazione dei `.sync` servizi, Step Functions utilizza il polling che utilizza la quota e gli eventi assegnati per monitorare lo stato di un lavoro. Per `.sync` le chiamate all'interno dello stesso account, Step Functions utilizza EventBridge eventi ed esegue il polling delle API specificate nello stato. `Task` Per le chiamate tra più account`.sync`, Step Functions utilizza solo il polling. Ad esempio, for`states:StartExecution.sync`, Step Functions esegue il polling sull'DescribeExecutionAPI e utilizza la quota assegnata.

> **Tip**
>
> Per distribuire un flusso di lavoro di esempio che utilizza il modello di integrazione del servizio Run a Job (.sync) sul tuo Account AWS, consulta il Modulo 3 - Run a Job (.sync) di The Workshop. AWS Step Functions

Per consultare un elenco di ciò che supportano i servizi integrati in attesa del completamento di un processo (`.sync`), vedi Integrazioni ottimizzate per Step Functions.

> **Note**
>
> Le integrazioni di servizi che utilizzano il `.sync` pattern richiedono autorizzazioni IAM aggiuntive. Per ulteriori informazioni, consulta Politiche IAM per servizi integrati.

In alcuni casi, potresti volere che Step Functions continui il flusso di lavoro prima che il lavoro sia completamente completato. È possibile ottenere ciò nello stesso modo in cui si utilizza il modello Attendere un callback con il token dell'attività di integrazione dei servizi. Per fare ciò, passa un token di attività al tuo job, quindi restituiscilo utilizzando una chiamata `SendTaskSuccess`o `SendTaskFailure`API. Step Functions utilizzerà i dati forniti in quella chiamata per completare l'attività, interrompere il monitoraggio del lavoro e continuare il flusso di lavoro.

Attendere un callback con il token dell'attività

Le attività di callback forniscono un modo per mettere in pausa un flusso di lavoro finché non viene restituito il token dell'attività. Per un'operazione potrebbe essere necessario attendere un'approvazione umana, integrarsi con una terza parte oppure chiamare i sistemi legacy. Per attività come queste, puoi mettere in pausa Step Functions finché l'esecuzione del flusso di lavoro non raggiunge la quota di servizio di un anno (vedi,Quote relative alla limitazione statale) e attendere il completamento di un processo o flusso di lavoro esterno. In queste situazioni Step Functions consente di passare un token di attività alle integrazioni del servizio AWS SDK e anche ad alcune integrazioni di servizi ottimizzati. L'attività verrà messa in pausa finché non riceverà la restituzione del token con una chiamata `SendTaskSuccess` o `SendTaskFailure`.

Se uno `Task` stato che utilizza il token di attività di callback scade, viene generato un nuovo token casuale. È possibile accedere ai token delle attività dall'oggetto context.

> **Note**
>
> Un task token deve contenere almeno un carattere e non può superare i 1024 caratteri.

Per essere utilizzata `.waitForTaskToken` con un'integrazione AWS SDK, l'API utilizzata deve disporre di un campo di parametri in cui inserire il token dell'attività.

> **Note**
>
> È necessario trasferire i token delle attività dai responsabili all'interno dello stesso account. AWS I token non funzioneranno se li invii dai responsabili di un altro account. AWS

> **Tip**
>
> Per implementare un flusso di lavoro di esempio che utilizza un modello di integrazione del servizio di token di attività di callback Account AWS, consulta il Modulo 4 - Attendi una richiamata con il Task Token di The Workshop. AWS Step Functions

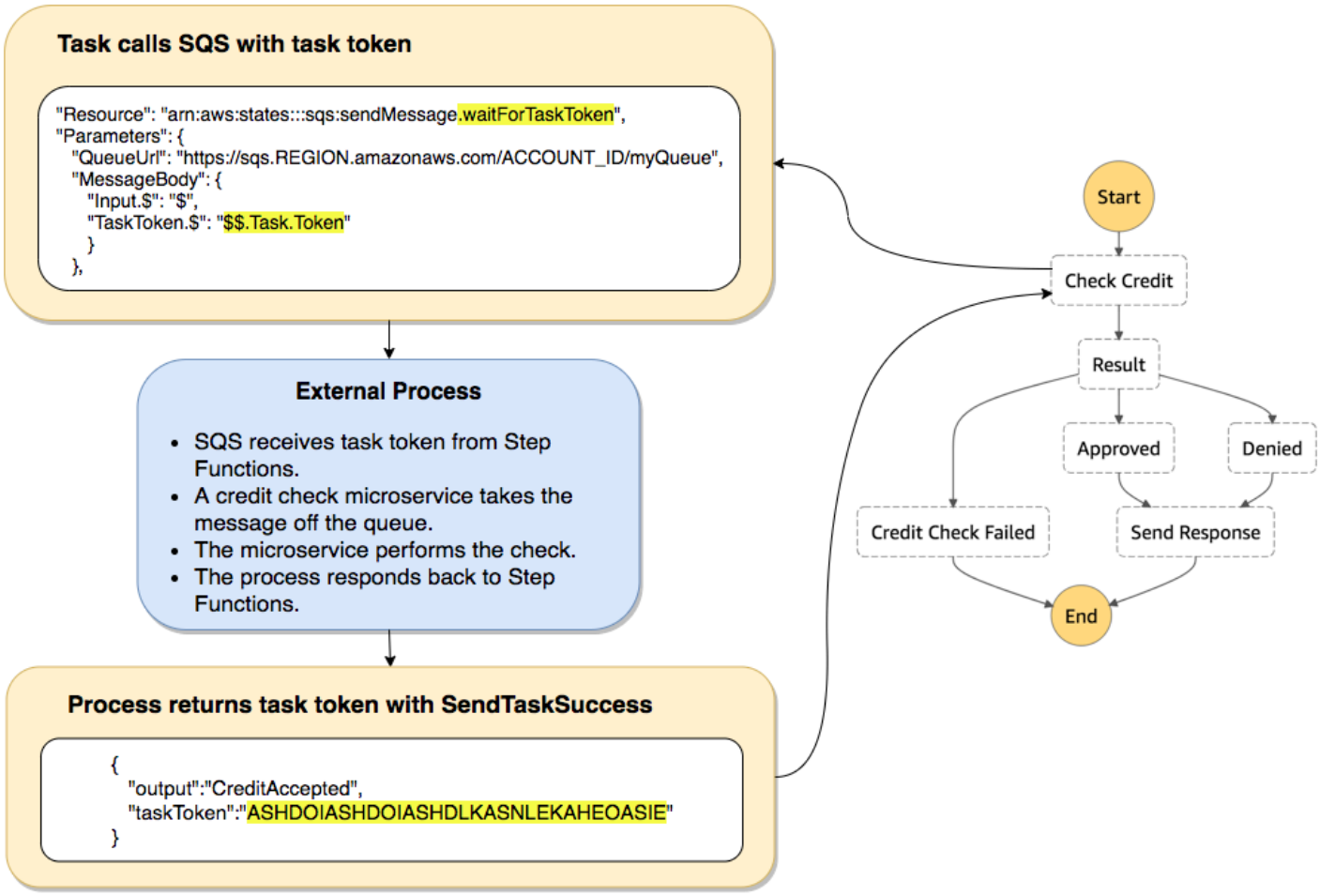
Per consultare un elenco di ciò che supportano i servizi integrati in attesa del token di un'attività (`.waitForTaskToken`), vedi Integrazioni ottimizzate per Step Functions.

Argomenti

- Esempio di token di un'attività
- Ottieni un Token dall'oggetto contestuale
- Configurazione di un timeout Heartbeat per un'operazione in attesa

Esempio di token di un'attività

In questo esempio, un flusso di lavoro Step Functions deve integrarsi con un microservizio esterno per eseguire una verifica del credito come parte di un flusso di lavoro di approvazione. Step Functions pubblica un messaggio Amazon SQS che include un task token come parte del messaggio. Un sistema esterno si integra con Amazon SQS e rimuove il messaggio dalla coda. Al termine, restituisce il risultato e il task token originale. Step Functions continua quindi con il suo flusso di lavoro.



Il `"Resource"` campo della definizione dell'attività che fa riferimento ad Amazon SQS include `.waitForTaskToken` aggiunto alla fine.

```
"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"
}
```

Questo dice a Step Functions di mettere in pausa e attendere il task token. Quando si specifica una risorsa utilizzando `.waitForTaskToken`, il token di attività è accessibile nel campo `"Parameters"` della definizione dello stato con una designazione speciale di percorso (`$$.Task.Token`). Il `$$.` iniziale indica che il percorso accede all'oggetto contestuale e ottiene il token di attività per l'attività corrente in un'esecuzione in corso.

Quando viene completato, il servizio esterno chiama `SendTaskSuccess` o `SendTaskFailure` con `taskToken` incluso. Solo allora il flusso di lavoro procede allo stato successivo.

> **Note**
>
> Per evitare un'attesa indefinita se un processo non riesce a inviare il token di attività insieme a `SendTaskSuccess` o `SendTaskFailure`, consulta Configurazione di un timeout Heartbeat per un'operazione in attesa.

Ottieni un Token dall'oggetto contestuale

L'oggetto contestuale è un oggetto JSON interno che contiene informazioni sull'esecuzione. Come input di stato, è possibile accedervi con un percorso dal campo `"Parameters"` durante un'esecuzione. Quando l'accesso viene eseguito dall'interno di una definizione di attività, include informazioni sull'esecuzione specifica, tra cui il token dell'attività.

```
{
    "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"
    }
}
```

È possibile accedere al token dell'attività utilizzando un percorso speciale dall'interno del campo `"Parameters"` della definizione dell'attività. Per accedere all'input o all'oggetto contestuale, specificare per prima cosa che il parametro sarà un percorso aggiungendo `.$` al nome del parametro. Quanto segue specifica i nodi sia dall'input sia dall'oggetto contestuale in una specifica `"Parameters"`.

```
"Parameters": {
   "Input.$": "$",
   "TaskToken.$": "$$.Task.Token"
 },
```

In entrambi i casi, l'aggiunta `.$` al nome del parametro indica a Step Functions di aspettarsi un percorso. Nel primo caso, `"$"` è un percorso che include l'intero input. Nel secondo caso, `$$.` specifica che il percorso potrà accedere all'oggetto contestuale e `$$.Task.Token` imposta il parametro al valore del token di attività nell'oggetto contestuale di un'esecuzione.

Nell'esempio di Amazon SQS, `.waitForTaskToken` nel `"Resource"` campo indica a Step Functions di attendere la restituzione del token dell'attività. Il `"TaskToken.$": "`**`$$.Task.Token`**`"` parametro passa quel token come parte del messaggio 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"
}
```

Per ulteriori informazioni sugli oggetti contesto, vedi Oggetto Context nella sezione Elaborazione di input e output di questa guida.

Configurazione di un timeout Heartbeat per un'operazione in attesa

Un'operazione in attesa del token di attività attenderà che l'esecuzione raggiunga la quota di servizio di un anno (vedi Quote relative alla limitazione statale). Per evitare il blocco delle esecuzioni è possibile configurare un intervallo di timeout heartbeat nella definizione della macchina a stati. Utilizzare il campo `HeartbeatSeconds` per specificare l'intervallo di timeout.

```
{
  "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
    }
  }
}
```

In questa definizione di macchina a stati, un'attività invia un messaggio ad Amazon SQS e attende che un processo esterno richiami il token di attività fornito. Il campo `"HeartbeatSeconds": 600` imposta l'intervallo di timeout heartbeat su 10 minuti. L'operazione attenderà che il token dell'attività venga restituito con una di queste azioni API:

- `SendTaskSuccess`
- `SendTaskFailure`
- `SendTaskHeartbeat`

Se l'attività di attesa non riceve un token di attività valido in un periodo di 10 minuti, l'operazione ha esito negativo con nome di errore `States.Timeout`.

Per ulteriori informazioni, consulta il progetto di esempio dell'attività di callback: Esempio di pattern di callback (Amazon SQS, Amazon SNS, Lambda) .

Passa i parametri a un'API di servizio

Utilizza il campo `Parameters` in uno stato `Task` per controllare quali parametri trasferire a un'API di servizio.

All'interno del `Parameters` campo, è necessario utilizzare la forma plurale dei parametri dell'array in un'azione API. Ad esempio, se utilizzi il campo Filtro dell'azione `DescribeSnapshots` API per l'integrazione con Amazon EC2, devi definire il campo come. `Filters` Se non si utilizza la forma plurale, Step Functions restituisce il seguente errore:

```
The field Filter is not supported by Step Functions.
```

Passa JSON statico come parametri

Puoi includere un oggetto JSON nella definizione della tua macchina a stati in modo da trasferire un parametro a una risorsa.

Ad esempio, per impostare il parametro `RetryStrategy` dell'API `SubmitJob` per AWS Batch, puoi includere quanto segue nei parametri.

```
"RetryStrategy": {
  "attempts": 5
}
```

Con gli oggetti JSON statici puoi inoltre trasmettere più parametri. Come esempio più completo, di seguito sono riportati i `Parameters` campi `Resource` e delle specifiche di un'attività che viene pubblicata su un argomento Amazon SNS denominato. *`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"
      }
    }
  },
```

Passa l'input di stato come parametri usando Paths

È possibile passare porzioni dell'input di stato come parametri utilizzando i percorsi. Un percorso è una stringa, a partire da$, utilizzata per identificare i componenti all'interno del testo JSON. I percorsi di Step Functions utilizzano la JsonPathsintassi.

Per specificare che un parametro utilizza un percorso, termina il nome del parametro con`.$`. Ad esempio, se l'input di stato contiene testo all'interno di un nodo denominato`message`, è possibile passare quel testo come parametro utilizzando un percorso.

Considerate il seguente input di stato:

```
{
  "comment": "A message in the state input",
  "input": {
    "message": "foo",
    "otherInfo": "bar"
  },
  "data": "example"
}
```

Per passare il valore del nodo denominato `message` come parametro, specificate la seguente sintassi:

```
"Parameters": {"myMessage.$": "$.input.message"},
```

Step Functions passa quindi il valore `foo` come parametro.

Per ulteriori informazioni sull'utilizzo dei parametri in Step Functions, vedere quanto segue:

- Elaborazione di input e output
- InputPath, Parametri e ResultSelector

Passa nodi di oggetti contestuali come parametri

Oltre a contenuti statici e nodi dall'input di stato, è possibile trasferire i nodi dall'oggetto contestuale come parametri. L'oggetto contestuale è un dato JSON dinamico che esiste durante l'esecuzione di una macchina di stato. Include informazioni sulla tua macchina di stato e l'esecuzione corrente. È possibile accedere all'oggetto contestuale utilizzando un percorso nel campo `"Parameters"` di una definizione dello stato.

Per ulteriori informazioni sull'oggetto contesto e su come accedere a tali dati da un campo `"Parameters"`, consulta:

- Oggetto Context
- Accesso all'oggetto contestuale
- Ottieni un Token dall'oggetto contestuale

Registro delle modifiche per le integrazioni AWS SDK supportate

La tabella seguente riassume quando i servizi sono inizialmente integrati con Step Functions e quando la loro API di integrazione è stata aggiornata più di recente. Per i dettagli sull'utilizzo delle integrazioni, consulta. AWS Integrazioni di servizi SDK

> **Important**
>
> Il supporto alle azioni API viene rilasciato con cadenza trimestrale. Gli aggiornamenti alle azioni già supportate, come i nuovi parametri, potrebbero non essere immediatamente disponibili.

Servizio	Supporto iniziale	Aggiornato	
AWS AppFabric	18 gennaio 2024		
B2B Data Interchange	18 gennaio 2024		
Esportazioni di dati AWS	18 gennaio 2024		

Servizio	Supporto iniziale	Aggiornato	
Amazon Bedrock	18 gennaio 2024		
Amazon Bedrock Agents	18 gennaio 2024		
Amazon Bedrock Runtime Agents	18 gennaio 2024		
Amazon Bedrock Runtime	18 gennaio 2024		
Amazon CloudFront KeyValueStore	18 gennaio 2024		
Amazon CodeGuru Security	18 gennaio 2024		
Centrale ottimizza zione costi AWS	18 gennaio 2024		
Amazon DataZone	18 gennaio 2024		
Amazon EKS Auth	18 gennaio 2024		
AWS Entity Resolutio n	18 gennaio 2024		
Piano gratuito di AWS	18 gennaio 2024		
Amazon Inspector Scan	18 gennaio 2024		
AWS Launch Wizard	18 gennaio 2024		
Amazon Managed Blockchain Query	18 gennaio 2024		

Servizio	Supporto iniziale	Aggiornato	
AWS Elemental MediaPackage V2	18 gennaio 2024		
AWS HealthImaging	18 gennaio 2024		
Network Manager	18 gennaio 2024		
AWS Payment Cryptography	18 gennaio 2024		
AWS Payment Cryptography Data	18 gennaio 2024		
AWS Private CA Connector for Active Directory	18 gennaio 2024		
Amazon Q Business	18 gennaio 2024		
Amazon Q Connect	18 gennaio 2024		
AWS re:Post	18 gennaio 2024		
Amazon Timestream Query	18 gennaio 2024		
Amazon Timestream Write	18 gennaio 2024		
Trusted Advisor	18 gennaio 2024		
Verified Permissions	18 gennaio 2024		
Amazon WorkSpaces Thin Client	18 gennaio 2024		
AWS CloudTrail Data	16 giugno 2023		

Servizio	Supporto iniziale	Aggiornato	
Amazon CloudWatch Internet Monitor	16 giugno 2023		
Amazon Interacti ve Video Service RealTime	16 giugno 2023		
AWS IoT TwinMaker	16 giugno 2023		
Amazon OpenSearch Ingestion	16 giugno 2023		
AWS Telco Network Builder	16 giugno 2023		
Amazon VPC Lattice	16 giugno 2023		
AWS Backup Storage	17 febbraio 2023		
Amazon Chime Media Pipelines	17 febbraio 2023		
Amazon Chime Voice	17 febbraio 2023		
AWS Clean Rooms	17 febbraio 2023	18 gennaio 2024	
Amazon CodeCatalyst	17 febbraio 2023		
Amazon Connect Cases	17 febbraio 2023		
AWS Control Tower	17 febbraio 2023		
Amazon DocumentDB Elastic Clusters	17 febbraio 2023		
Amazon EMR Serverless	17 febbraio 2023		

Servizio	Supporto iniziale	Aggiornato	
Amazon IVS Chat	17 febbraio 2023		
Amazon Kendra Intelligent Ranking	17 febbraio 2023		
AWS HealthOmics	17 febbraio 2023		
Amazon Redshift Serverless	17 febbraio 2023		
Amazon Security Lake	17 febbraio 2023		
AWS Health	17 febbraio 2023		
AWS IoT FleetWise	17 febbraio 2023		
AWS IoT RoboRunner	17 febbraio 2023		
AWS Mainframe Modernization	17 febbraio 2023		
Orchestratore dell'Hub di migrazione AWS	17 febbraio 2023		
AWS Private 5G	17 febbraio 2023		
Esploratore di risorse AWS	17 febbraio 2023		
AWS SimSpace Weaver	17 febbraio 2023		
AWS Support App	17 febbraio 2023		

Servizio	Supporto iniziale	Aggiornato	
CloudWatch Observability Access Manager	17 febbraio 2023		
EventBridge Pipes	17 febbraio 2023		
EventBridge Scheduler	17 febbraio 2023		
IAM Roles Anywhere	17 febbraio 2023		
Kinesis Video WebRTC Storage	17 febbraio 2023		
License Manager Linux Subscriptions	17 febbraio 2023		
License Manager User Subscriptions	17 febbraio 2023		
OpenSearch Serverless	17 febbraio 2023		
Route 53 ARC Zonal Shift	17 febbraio 2023		
SageMaker Geospatial	17 febbraio 2023		
SageMaker Metrics	17 febbraio 2023		
Systems Manager for SAP	17 febbraio 2023		
AWS Account Management	19 aprile 2022		
AWS Amplify	30 settembre 2021		

Servizio	Supporto iniziale	Aggiornato	
AWS App Mesh	30 settembre 2021		
AWS App Runner	30 settembre 2021	17 febbraio 2023	
AWS AppConfig	30 settembre 2021		
AWS AppConfig Data	19 aprile 2022		
AWS AppSync	30 settembre 2021	17 febbraio 2023	
AWS Application Discovery Service	30 settembre 2021		
AWS Application Migration Service	30 settembre 2021		
AWS Audit Manager	30 settembre 2021		
AWS Auto Scaling Plans	30 settembre 2021		
AWS Backup	30 settembre 2021	17 febbraio 2023	
AWS Backup gateway	19 aprile 2022	17 febbraio 2023	
AWS Batch	30 settembre 2021	17 febbraio 2023	
AWS Billing Conductor	26 luglio 2022	17 febbraio 2023	
Budget AWS	30 settembre 2021		
AWS Certificate Manager	30 settembre 2021		
AWS Private Certificate Authority	30 settembre 2021		
AWS Cloud Map	30 settembre 2021		

Servizio	Supporto iniziale	Aggiornato	
AWS Cloud9	30 settembre 2021		
AWS CloudFormation	30 settembre 2021	17 febbraio 2023	
AWS CloudHSM	30 settembre 2021		
AWS CloudHSM	30 settembre 2021		
AWS CloudTrail	30 settembre 2021	17 febbraio 2023	
AWS Cloud Control	19 aprile 2022		
AWS CodeBuild	30 settembre 2021		
AWS CodeCommit	30 settembre 2021	17 febbraio 2023	
AWS CodeDeploy	30 settembre 2021		
AWS CodePipeline	30 settembre 2021		
AWS CodeStar	30 settembre 2021		
AWS CodeStar	30 settembre 2021		
AWS CodeStar	30 settembre 2021		
AWS Compute Optimizer	30 settembre 2021	17 febbraio 2023	
AWS Config	30 settembre 2021	26 luglio 2022	
AWS Cost Explorer Service	30 settembre 2021	17 febbraio 2023	
AWS Cost and Usage Report	30 settembre 2021		
AWS Data Exchange	30 settembre 2021	26 luglio 2022	
AWS Data Pipeline	30 settembre 2021		

Servizio	Supporto iniziale	Aggiornato	
AWS DataSync	30 settembre 2021	26 luglio 2022	
AWS Database Migration Service	30 settembre 2021		
AWS Device Farm	30 settembre 2021		
AWS Direct Connect	30 settembre 2021		
AWS Directory Service	30 settembre 2021	17 febbraio 2023	
AWS EC2 Instance Connect	30 settembre 2021		
AWS Elastic Beanstalk	30 settembre 2021		
AWS Elemental MediaLive	30 settembre 2021		
AWS Elemental MediaPackage	30 settembre 2021		
AWS Elemental MediaPackage VOD	30 settembre 2021		
AWS Elemental MediaStore	30 settembre 2021		
AWS Fault Injection Service	30 settembre 2021		
AWS Firewall Manager	30 settembre 2021	17 febbraio 2023	
AWS Glue	30 settembre 2021	17 febbraio 2023	

Servizio	Supporto iniziale	Aggiornato	
AWS Glue DataBrew	30 settembre 2021		
AWS IoT Greengrass	30 settembre 2021		
AWS Ground Station	30 settembre 2021	17 febbraio 2023	
AWS Identity and Access Management	30 settembre 2021		
AWS IoT	30 settembre 2021	17 febbraio 2023	
AWS IoT 1-Click	30 settembre 2021		
AWS IoT Analytics	30 settembre 2021		
AWS IoT Core Device Advisor	30 settembre 2021		
AWS IoT Events	30 settembre 2021		
Dati Eventi AWS IoT	30 settembre 2021		
AWS IoT Fleet Hub	30 settembre 2021		
AWS IoT Greengrass Version 2	30 settembre 2021		
AWS IoT Jobs Data Plane	30 settembre 2021		
AWS IoT Secure Tunneling	30 settembre 2021		
AWS IoT SiteWise	30 settembre 2021	17 febbraio 2023	
AWS IoT Things Graph	30 settembre 2021		
AWS IoT Wireless	30 settembre 2021	17 febbraio 2023	

Servizio	Supporto iniziale	Aggiornato	
AWS Key Management Service	30 settembre 2021	26 luglio 2022	
AWS Lake Formation	30 settembre 2021	17 febbraio 2023	
AWS Lambda	30 settembre 2021	17 febbraio 2023	
AWS License Manager	30 settembre 2021	17 febbraio 2023	
Marketplace AWS	30 settembre 2021	17 febbraio 2023	
AWS Marketplace Commerce Analytics	30 settembre 2021		
AWS Marketplace Entitlement Service	30 settembre 2021		
AWS Elemental MediaTailor	30 settembre 2021	26 luglio 2022	
AWS Migration Hub	30 settembre 2021		
AWS Migration Hub Config	30 settembre 2021		
Suggerimenti di strategia dell'Hub di migrazione AWS	19 aprile 2022	17 febbraio 2023	
AWS Mobile	30 settembre 2021		
AWS Network Firewall	30 settembre 2021		
AWS OpsWorks	30 settembre 2021		
AWS OpsWorks CM	30 settembre 2021		
AWS Organizations	30 settembre 2021	17 febbraio 2023	

Servizio	Supporto iniziale	Aggiornato	
AWS Outposts	30 settembre 2021		
AWS Panorama	19 aprile 2022	17 febbraio 2023	
Amazon Relationa l Database Service Performance Insights	30 settembre 2021		
Listino prezzi AWS	30 settembre 2021		
Amazon Relational Database Service	30 settembre 2021		
AWS Resilience Hub	19 aprile 2022		
AWS Resource Access Manager	30 settembre 2021		
AWS Resource Groups	30 settembre 2021	17 febbraio 2023	
AWS Resource Groups Tagging API	30 settembre 2021		
AWS RoboMaker	30 settembre 2021		
AWS IAM Identity Center	30 settembre 2021	17 febbraio 2023	
AWS SSO OIDC	30 settembre 2021		
AWS Secrets Manager	30 settembre 2021		
AWS Security Token Service	30 settembre 2021		
AWS Security Hub	30 settembre 2021		

Servizio	Supporto iniziale	Aggiornato	
AWS Server Migration Service	30 settembre 2021		
AWS Service Catalog	30 settembre 2021		
AWS Service Catalog AppRegistry	30 settembre 2021	17 febbraio 2023	
AWS Shield	30 settembre 2021		
AWS Signer	30 settembre 2021		
AWS IAM Identity Center	30 settembre 2021		
AWS IAM Identity Center Admin	30 settembre 2021		
AWS Step Functions	30 settembre 2021	17 febbraio 2023	
AWS Storage Gateway	30 settembre 2021		
AWS Support	30 settembre 2021		
AWS Transfer Family	30 settembre 2021	17 febbraio 2023	
AWS WAF	30 settembre 2021		
AWS WAF Regional	30 settembre 2021		
AWS WAFV2	30 settembre 2021		
AWS Well-Architected Tool	30 settembre 2021	17 febbraio 2023	
AWS X-Ray	30 settembre 2021	17 febbraio 2023	

Servizio	Supporto iniziale	Aggiornato	
AWS Marketplace Metering Service	30 settembre 2021		
AWS Serverless Application Repository	30 settembre 2021		
AWS Identity and Access Management Access Analyzer	30 settembre 2021		
Alexa for Business	30 settembre 2021		
Amazon API Gateway	30 settembre 2021	17 febbraio 2023	
Amazon API Gateway	30 settembre 2021		
Amazon AppIntegrations	30 settembre 2021		
Amazon AppStream 2.0	30 settembre 2021		
Amazon AppFlow	30 settembre 2021	17 febbraio 2023	
Amazon Athena	30 settembre 2021	17 febbraio 2023	
Amazon Augmented AI	30 settembre 2021		
Amazon Braket	30 settembre 2021		
Amazon Chime	30 settembre 2021		
Amazon Chime Meetings	19 aprile 2022	17 febbraio 2023	
Amazon Cloud Directory	30 settembre 2021		

Servizio	Supporto iniziale	Aggiornato	
Amazon CloudFront	30 settembre 2021	17 febbraio 2023	
Amazon CloudSearch	30 settembre 2021		
Amazon CloudWatch	30 settembre 2021	17 febbraio 2023	
Amazon CloudWatch Application Insights	30 settembre 2021		
CloudWatch Evidently	19 aprile 2022		
Amazon CloudWatch Logs	30 settembre 2021		
Amazon CloudWatch RUM	19 aprile 2022	17 febbraio 2023	
Amazon CloudWatch Synthetics	30 settembre 2021		
Amazon CodeGuru Profiler	30 settembre 2021		
Amazon CodeGuru Reviewer	30 settembre 2021		
Amazon Cognito	30 settembre 2021		
Amazon Cognito Identity Provider	30 settembre 2021		
Amazon Cognito Sync	30 settembre 2021		
Amazon Comprehend	30 settembre 2021	17 febbraio 2023	
Amazon Comprehend Medical	30 settembre 2021		

Servizio	Supporto iniziale	Aggiornato	
Amazon Connect Contact Lens	30 settembre 2021		
Amazon Connect Participant Service	30 settembre 2021		
Amazon Connect	30 settembre 2021	17 febbraio 2023	
Amazon Connect Voice ID	19 aprile 2022		
Amazon Connect Wisdom	19 aprile 2022		
Amazon Data Lifecycle Manager	30 settembre 2021		
Amazon Detective	30 settembre 2021		
Amazon DevOps Guru	30 settembre 2021	26 luglio 2022	
Amazon DocumentDB (with MongoDB compatibility)	30 settembre 2021		
Amazon DynamoDB	30 settembre 2021	17 febbraio 2023	
Amazon DynamoDB Streams	30 settembre 2021		
Amazon EC2 Container Registry	30 settembre 2021		
Amazon EC2 Container Service	30 settembre 2021	17 febbraio 2023	

Servizio	Supporto iniziale	Aggiornato	
Amazon EC2 Systems Manager	30 settembre 2021	17 febbraio 2023	
Amazon EMR	30 settembre 2021	17 febbraio 2023	
Amazon ElastiCache	30 settembre 2021		
Amazon Elastic Inference	30 settembre 2021		
Amazon Elastic Block Store	30 settembre 2021		
Amazon Elastic Compute Cloud	30 settembre 2021	17 febbraio 2023	
Amazon Elastic Container Registry Public	30 settembre 2021		
Amazon Elastic File System	30 settembre 2021		
Amazon Elastic Kubernetes Service	30 settembre 2021	17 febbraio 2023	
Amazon EMR	30 settembre 2021		
Amazon Elastic Transcoder	30 settembre 2021		
Amazon OpenSearch Service	30 settembre 2021	17 febbraio 2023	
Amazon OpenSearch Service	19 aprile 2022	17 febbraio 2023	
Amazon EventBridge	30 settembre 2021	17 febbraio 2023	

Servizio	Supporto iniziale	Aggiornato	
Amazon FSx	30 settembre 2021	17 febbraio 2023	
Amazon Forecast Query	30 settembre 2021	17 febbraio 2023	
Amazon Forecast Service	30 settembre 2021	17 febbraio 2023	
Amazon Fraud Detector	30 settembre 2021		
Amazon GameLift	30 settembre 2021	17 febbraio 2023	
Amazon GameSparks	26 luglio 2022		
Amazon S3 Glacier	30 settembre 2021		
Amazon GuardDuty	30 settembre 2021		
AWS HealthLake	30 settembre 2021		
Amazon Honeycode	30 settembre 2021		
Amazon Inspector	30 settembre 2021		
Amazon Inspector V2	19 aprile 2022		
Amazon Interactive Video Service	30 settembre 2021		
Amazon Kendra	30 settembre 2021		
Amazon Kinesis	30 settembre 2021		
Amazon Kinesis Analytics	30 settembre 2021		

Servizio	Supporto iniziale	Aggiornato	
Amazon Kinesis Analytics V2	30 settembre 2021		
Amazon Kinesis Firehose	30 settembre 2021		
Amazon Kinesis Video Signaling Channels	30 settembre 2021		
Amazon Kinesis Video Streams	30 settembre 2021	17 febbraio 2023	
Amazon Kinesis Video Streams Archived Media	30 settembre 2021		
Amazon Kinesis video stream	30 settembre 2021		
Amazon Lex Model Building Service	30 settembre 2021		
Amazon Lex Model Building Service V2	30 settembre 2021	17 febbraio 2023	
Amazon Lex	30 settembre 2021		
Amazon Lex Runtime V2	30 settembre 2021		
Amazon Lightsail	30 settembre 2021	17 febbraio 2023	
Servizio di posizione Amazon	30 settembre 2021	17 febbraio 2023	
Amazon Lookout for Equipment	30 settembre 2021		

Servizio	Supporto iniziale	Aggiornato	
Amazon Lookout for Metrics	30 settembre 2021	17 febbraio 2023	
Amazon Lookout for Vision	30 settembre 2021		
Amazon MQ	30 settembre 2021		
Amazon Macie	30 settembre 2021		
Amazon Macie 2	30 settembre 2021	17 febbraio 2023	
Amazon Managed Blockchain	30 settembre 2021	17 febbraio 2023	
Amazon Managed Grafana	19 aprile 2022	17 febbraio 2023	
Amazon Managed Service for Prometheus	30 settembre 2021	17 febbraio 2023	
Amazon Managed Streaming for Apache Kafka	30 settembre 2021	17 febbraio 2023	
Amazon Managed Streaming for Apache Kafkax	19 aprile 2022		
Amazon Managed Workflows for Apache Airflow	30 settembre 2021		
Amazon Mechanical Turk	30 settembre 2021		

Servizio	Supporto iniziale	Aggiornato	
Amazon MemoryDB for Redis	19 aprile 2022	17 febbraio 2023	
Amazon Nimble Studio	30 settembre 2021		
Amazon Personalize	30 settembre 2021	17 febbraio 2023	
Amazon Personalize Events	30 settembre 2021		
Amazon Personalize Runtime	30 settembre 2021		
Amazon Pinpoint	30 settembre 2021		
Amazon Pinpoint Email Service	30 settembre 2021		
Amazon Pinpoint SMS and Voice Service	30 settembre 2021		
Amazon Pinpoint SMS and Voice V2 Service	26 luglio 2022		
Amazon Polly	30 settembre 2021		
Amazon QLDB	30 settembre 2021		
Amazon QLDB Session	30 settembre 2021		
Amazon QuickSight	30 settembre 2021	17 febbraio 2023	
Amazon Redshift	30 settembre 2021		

Servizio	Supporto iniziale	Aggiornato	
Amazon Redshift Data API	30 settembre 2021		
Amazon Rekognition	30 settembre 2021	17 febbraio 2023	
Amazon Relational Database Service	30 settembre 2021	17 febbraio 2023	
Amazon Route 53	30 settembre 2021		
Amazon Route 53 Recovery Control Config	30 settembre 2021	26 luglio 2022	
Amazon Route 53 Domains	30 settembre 2021	17 febbraio 2023	
Amazon Route 53 Resolver	30 settembre 2021		
Amazon S3 on Outposts	30 settembre 2021	26 luglio 2022	
Amazon SageMaker Runtime Feature Store Runtime	30 settembre 2021		
Amazon SageMaker Runtime Runtime	30 settembre 2021		
Amazon SageMaker	30 settembre 2021	17 febbraio 2023	
Amazon SageMaker Edge Manager	30 settembre 2021		
Amazon Simple Email Service	30 settembre 2021		

Servizio	Supporto iniziale	Aggiornato	
Amazon Simple Email Service V2	30 settembre 2021	17 febbraio 2023	
Amazon Simple Notification Service	30 settembre 2021	17 febbraio 2023	
Amazon Simple Queue Service	30 settembre 2021	17 febbraio 2023	
Amazon Simple Storage Service	30 settembre 2021	17 febbraio 2023	
Amazon Simple Workflow Service	30 settembre 2021		
Amazon Textract	30 settembre 2021	17 febbraio 2023	
Amazon Transcribe	30 settembre 2021		
Amazon Translate	30 settembre 2021	17 febbraio 2023	
Amazon WorkDocs	30 settembre 2021	17 febbraio 2023	
Amazon WorkMail	30 settembre 2021	17 febbraio 2023	
Amazon WorkMail Message Flow	30 settembre 2021		
Amazon WorkSpaces	30 settembre 2021	17 febbraio 2023	
Amazon WorkSpaces Web	19 aprile 2022	17 febbraio 2023	
Amplify	30 settembre 2021		
Amplify UI Builder	19 aprile 2022	17 febbraio 2023	
Application Auto Scaling	30 settembre 2021		

Servizio	Supporto iniziale	Aggiornato	
Amazon EC2 Auto Scaling	30 settembre 2021	17 febbraio 2023	
CodeArtifact	30 settembre 2021		
DynamoDB Accelerator	30 settembre 2021		
EC2 Image Builder	30 settembre 2021		
AWS Elastic Disaster Recovery	19 aprile 2022	17 febbraio 2023	
Elastic Load Balancing	30 settembre 2021		
Elastic Load Balancing V2	30 settembre 2021		
MediaConnect	30 settembre 2021		
Amazon S3 Control	30 settembre 2021	17 febbraio 2023	
Recycle Bin for Amazon EBS	19 aprile 2022	17 febbraio 2023	
Savings Plans	30 settembre 2021		
Amazon EventBridge Schema Registry	30 settembre 2021		
Service Quotas	30 settembre 2021		
AWS Snowball	30 settembre 2021		

Progetti di esempio per Step Functions

Nella AWS Step Functionsconsole, puoi scegliere uno dei seguenti modelli di avvio per distribuire macchine a stati sui tuoi. Account AWS Questi modelli iniziali sono progetti di ready-to-run esempio che creano automaticamente il prototipo e la definizione del flusso di lavoro e tutte le risorse correlate per il progetto. AWS

È possibile utilizzare questi progetti di esempio per distribuirli ed eseguirli così come sono oppure utilizzare i prototipi del flusso di lavoro per svilupparli. Se si basa su questi progetti, Step Functions crea il prototipo del flusso di lavoro, ma non distribuisce le risorse elencate nella definizione del flusso di lavoro.

Quando si distribuiscono i progetti di esempio, questi forniscono una macchina a stati completamente funzionante e creano le risorse correlate per l'esecuzione della macchina a stati. Quando si crea un progetto di esempio, Step Functions lo utilizza AWS CloudFormation per creare le risorse correlate a cui fa riferimento la macchina a stati.

Argomenti

- Gestire un processo batch (AWS Batch,Amazon SNS)
- Gestire un'attività relativa al contenitore (Amazon ECS,Amazon SNS)
- Trasferisci record di dati (Lambda,DynamoDB,Amazon SQS)
- Sondaggio per Job Status (AWS Batch Lambda,)
- Timer delle attività (Lambda, Amazon SNS)
- Esempio di pattern di callback (Amazon SQS, Amazon SNS, Lambda)
- Gestisci un lavoro in Amazon EMR
- Esegui un EMR Serverless lavoro
- Avvio di un workflow all'interno di un workflow (Step Functions, Lambda)
- Elabora dinamicamente i dati con uno stato della mappa
- Elabora un file CSV con Distributed Map
- Elabora i dati in un bucket Amazon S3 con Distributed Map
- Training su un modello di Machine Learning
- Ottimizzazione di un modello di Machine Learning.
- Elabora messaggi ad alto volume da Amazon SQS (Express Workflows)

- Esempio di checkpoint selettivo (flussi di lavoro Express)
- Crea un AWS CodeBuild progetto (CodeBuild, Amazon SNS)
- Preelabora i dati e addestra un modello di machine learning
- Esempio di orchestrazione Lambda
- Avvia una query su Athena
- Esegui più query (Amazon Athena, Amazon SNS)
- Interroga set di dati di grandi dimensioni (Amazon Athena, Amazon S3 e Amazon SNS AWS Glue)
- Mantieni aggiornati i dati (Amazon Athena, Amazon S3,) AWS Glue
- Gestisci un cluster Amazon EKS
- Effettuare una chiamata a API Gateway
- Chiama un microservizio in esecuzione su Fargate utilizzando l'integrazione API Gateway
- Invia un evento personalizzato a EventBridge
- Richiama flussi di lavoro Synchronous Express
- Esegui flussi di lavoro ETL/ELT utilizzando Amazon Redshift (Lambda, Amazon Redshift Data API)
- Utilizzo Step Functions e AWS Batch con gestione degli errori
- Trova un AWS Batch lavoro
- AWS Batch con Lambda
- Esegui il concatenamento dei prompt AI con Amazon Bedrock

Gestire un processo batch (AWS Batch,Amazon SNS)

Questo progetto di esempio illustra come inoltrare un processo in AWS Batch e, in base al suo esito, inviare una notifica Amazon SNS dedicata. Nel corso del progetto verranno creati una macchina a stati AWS Step Functions, un processo in AWS Batch e un argomento Amazon SNS.

Step Functions, in questa occasione, si avvale di una macchina a stati per invocare il processo in AWS Batch in modo sincrono. Ne attende l'esito, poi invia un argomento Amazon SNS con un messaggio in merito.

Fase 1: Creare la macchina a stati e fornire risorse

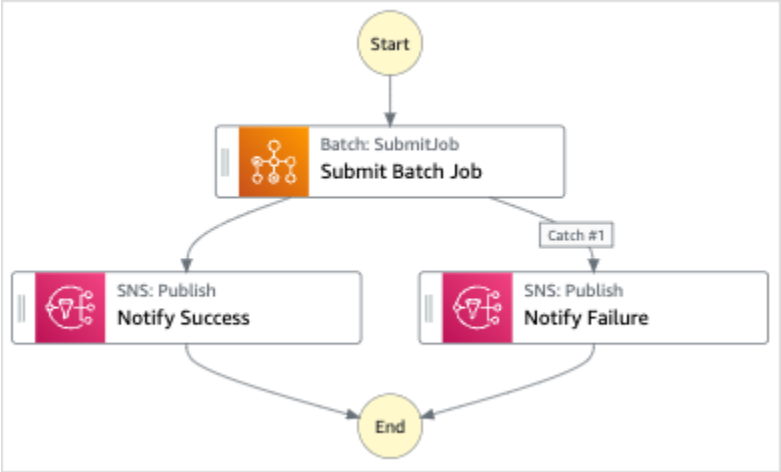
1. Apri la console Step Functions e scegli Crea macchina a stati.

2. Digita **`Manage a batch job`** nella casella di ricerca, quindi scegli Gestisci un processo in batch dai risultati della ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Questo progetto di esempio utilizza le seguenti risorse:

   - Un lavoro AWS Batch
   - Un argomento Amazon SNS
   - Una macchina AWS Step Functions statale
   - Ruoli correlati AWS Identity and Access Management (IAM)

   L'immagine seguente mostra il grafico del flusso di lavoro per il progetto di esempio Manage a batch job:



5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

     In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello

che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

> **Important**
>
> Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

- Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

  > **Tip**
  >
  > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

  Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

  La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Mentre le risorse vengono distribuite, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

  Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

  > **Important**
  >
  > Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.
2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

   1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

      > **Note**
      >
      > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

   2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

      Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

      > **Note**
      >
      > Se il progetto demo che hai distribuito contiene dati di input di esecuzione precompilati, usa quell'input per eseguire la macchina a stati.

   3. Selezionare Start execution (Avvia esecuzione).
   4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

      Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Esempio di codice della macchina a stati

La macchina a stati di questo progetto di esempio si integra con AWS Batch Amazon SNS passando i parametri direttamente a tali risorse.

Sfoglia questo esempio di macchina a stati per vedere come Step Functions controlla AWS Batch Amazon SNS connettendosi all'Amazon Resource Name (ARN) sul `Resource` campo e passando `Parameters` all'API del servizio.

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consulta. Utilizzo AWS Step Functions con altri servizi

```
{
  "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
    }
  }
}
```

Esempio IAM

Questo esempio di policy AWS Identity and Access Management (IAM) generato dal progetto di esempio include il minimo privilegio necessario per eseguire la macchina a stati e le risorse correlate. Ti consigliamo di includere solo le autorizzazioni necessarie nelle tue policy IAM.

```
{
    "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"
        }
    ]
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Gestire un'attività relativa al contenitore (Amazon ECS,Amazon SNS)

Questo progetto di esempio dimostra come eseguire un'AWS Fargateattività e quindi inviare una Amazon SNS notifica a seconda che il processo abbia esito positivo o negativo. La distribuzione di questo progetto di esempio creerà una macchina a AWS Step Functions stati, un Fargate cluster e un argomento. Amazon SNS

In questo progetto, Step Functions utilizza una macchina a stati per richiamare l'Fargateattività in modo sincrono. Ne attende l'esito, poi invia un argomento Amazon SNS con un messaggio in merito.

Fase 1: Creare la macchina a stati e fornire risorse

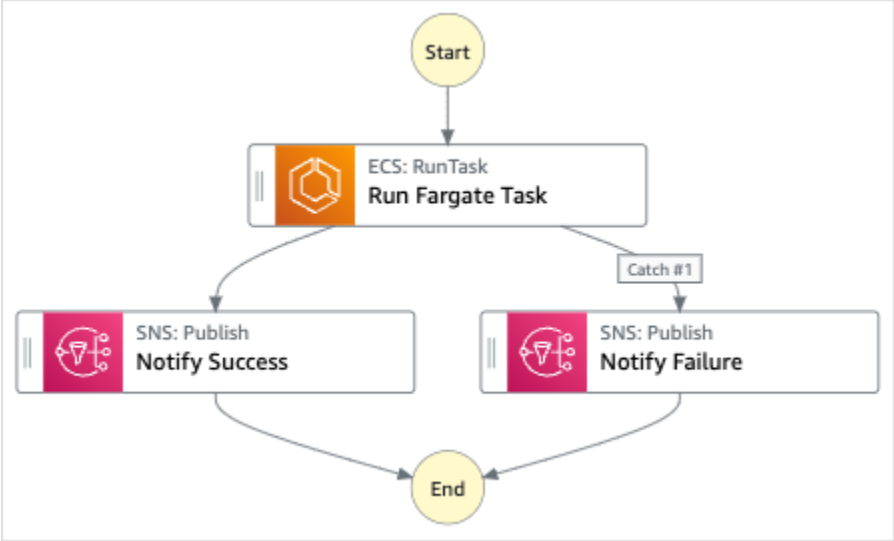
1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`Manage a container task`** nella casella di ricerca, quindi scegli Gestisci un'attività del contenitore dai risultati di ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul

tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

Questo progetto di esempio utilizza le seguenti risorse:

- Un cluster AWS Fargate
- Un argomento Amazon SNS
- Una macchina a AWS Step Functions stati
- Ruoli correlati AWS Identity and Access Management (IAM)

L'immagine seguente mostra il grafico del flusso di lavoro per il progetto di esempio Manage a Container Task:



5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

     In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

> **Important**
>
> Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

- Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

  > **Tip**
  >
  > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

  Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

  La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Mentre le risorse vengono distribuite, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

  Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

  > **Important**
  >
  > Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.
2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

   > **Note**
   >
   > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

   Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

   > **Note**
   >
   > Se il progetto demo che hai distribuito contiene dati di input di esecuzione precompilati, usa quell'input per eseguire la macchina a stati.

3. Selezionare Start execution (Avvia esecuzione).
4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

   Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Esempio di codice della macchina a stati

La macchina a stati di questo progetto di esempio si integra con AWS Fargate Amazon SNS passando i parametri direttamente a tali risorse. Sfoglia questo esempio di macchina a stati per

vedere come Step Functions utilizza una macchina a stati per chiamare l'attività Fargate in modo sincrono, attende che l'attività abbia esito positivo o negativo e invia un argomento di Amazon SNS con un messaggio sull'esito positivo o negativo del processo.

Per ulteriori informazioni su come controllare altri servizi, AWS Step Functions consulta. AWS Utilizzo AWS Step Functions con altri servizi

```
{
  "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
    }
  }
}
```

Esempio IAM

Questo esempio di policy AWS Identity and Access Management (IAM) generato dal progetto di esempio include il minimo privilegio necessario per eseguire la macchina a stati e le risorse correlate. È consigliabile includere solo le autorizzazioni necessarie nelle policy IAM.

```
{
    "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"
        }
    ]
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Trasferisci record di dati (Lambda,DynamoDB,Amazon SQS)

Questo progetto di esempio dimostra come leggere iterativamente gli elementi da una Amazon DynamoDB tabella e inviarli a una coda utilizzando una Amazon SQS macchina a stati. Step Functions L'implementazione di questo progetto di esempio creerà una macchina a Step Functions stati, una DynamoDB tabella, una AWS Lambda funzione e una coda. Amazon SQS

In questo progetto, Step Functions utilizza la Lambda funzione per popolare la tabella. DynamoDB La macchina a stati utilizza anche un `for` ciclo per leggere ciascuna voce e quindi invia ogni voce a una Amazon SQS coda.

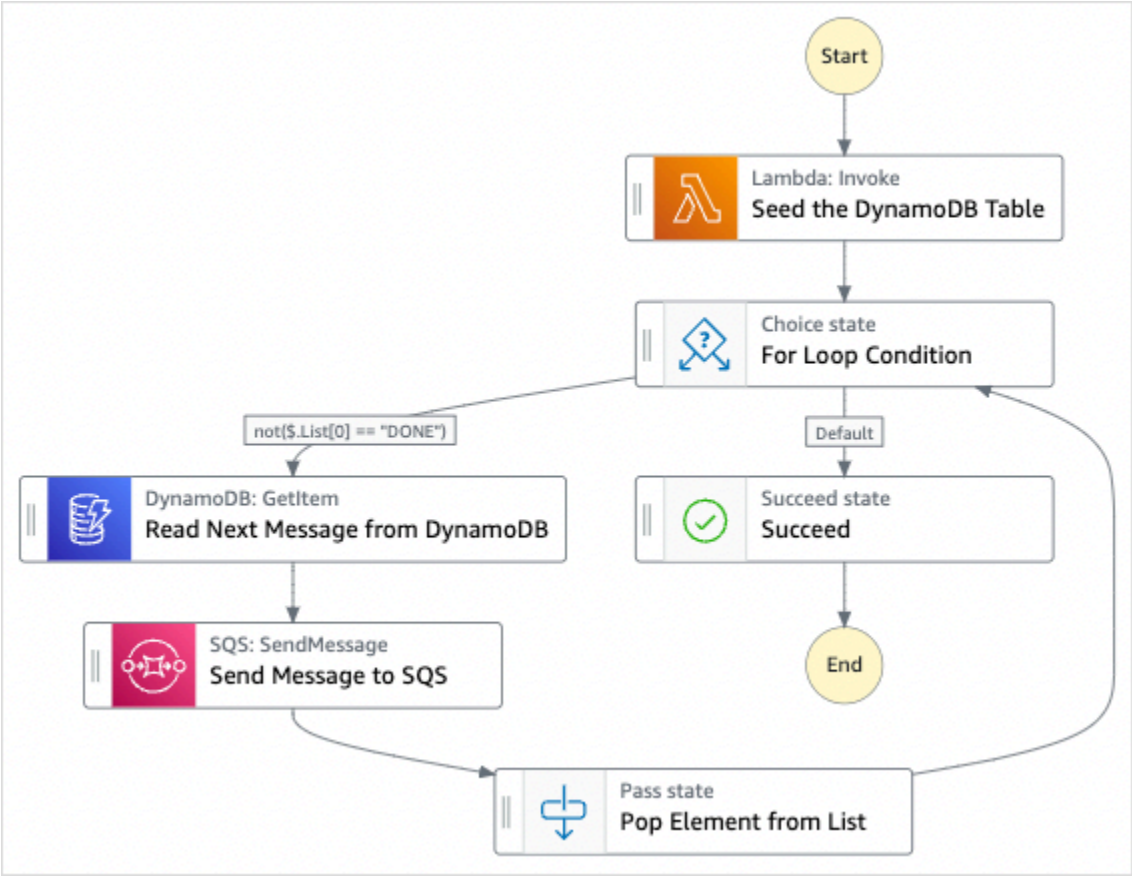
Fase 1: Creare la macchina a stati e fornire risorse

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`Transfer data records`** nella casella di ricerca, quindi scegli Trasferisci record di dati dai risultati della ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Questo progetto di esempio utilizza le seguenti risorse:

   - Una funzione Lambda per il seeding della tabella DynamoDB
   - Una coda Amazon SQS
   - Una tabella DynamoDB
   - AWS Step Functions Una macchina a stati
   - Ruoli correlati AWS Identity and Access Management (IAM)

   L'immagine seguente mostra il grafico del flusso di lavoro per il progetto di esempio Transfer data records:



5. Scegliete Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

     In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

     > **Important**
     >
     > Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

- Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

  > **Tip**
  >
  > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

  Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

  La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Mentre le risorse vengono distribuite, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

  Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

  > **Important**
  >
  > Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.
2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

   1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

> **Note**
>
> Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

   Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

   > **Note**
   >
   > Se il progetto demo che hai distribuito contiene dati di input di esecuzione precompilati, usa quell'input per eseguire la macchina a stati.

3. Selezionare Start execution (Avvia esecuzione).
4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

   Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Esempio di codice della macchina a stati

La macchina a stati di questo progetto di esempio si integra con DynamoDB e Amazon SQS passando i parametri direttamente a tali risorse.

Sfoglia questo esempio di macchina a stati per vedere come Step Functions controlla DynamoDB e Amazon SQS connettendosi all'Amazon Resource Name (ARN) sul campo e passando `Resource` all'API del servizio. `Parameters`

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consulta. Utilizzo AWS Step Functions con altri servizi

```
{
  "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"
    }
  }
}
```

Per ulteriori informazioni sul trasferimento di parametri e sulla gestione dei risultati, consultare i seguenti argomenti:

- Passa i parametri a un'API di servizio
- ResultPath

Esempio IAM

Questo esempio di policy AWS Identity and Access Management (IAM) generato dal progetto di esempio include il minimo privilegio necessario per eseguire la macchina a stati e le risorse correlate. È consigliabile includere solo le autorizzazioni necessarie nelle policy IAM.

```
{
    "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"
        }
    ]
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Sondaggio per Job Status (AWS Batch Lambda,)

Questo progetto di esempio crea un AWS Batch sondaggio sulle offerte di lavoro. Implementa una macchina a AWS Step Functions stati che utilizza AWS Lambda per creare un loop di `Wait` stato che controlla un AWS Batch lavoro.

Questo progetto di esempio crea e configura tutte le risorse in modo che il flusso di lavoro Step Functions invii un AWS Batch lavoro e attenda il completamento di tale processo prima di terminare correttamente.

> **Note**
>
> È inoltre possibile implementare questo modello senza utilizzare una funzione Lambda. Per informazioni sul controllo AWS Batch diretto, consultaUtilizzo AWS Step Functions con altri servizi.

Questo progetto di esempio crea la macchina a stati, due funzioni Lambda e una AWS Batch coda e configura le relative autorizzazioni IAM.

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consulta. Utilizzo AWS Step Functions con altri servizi

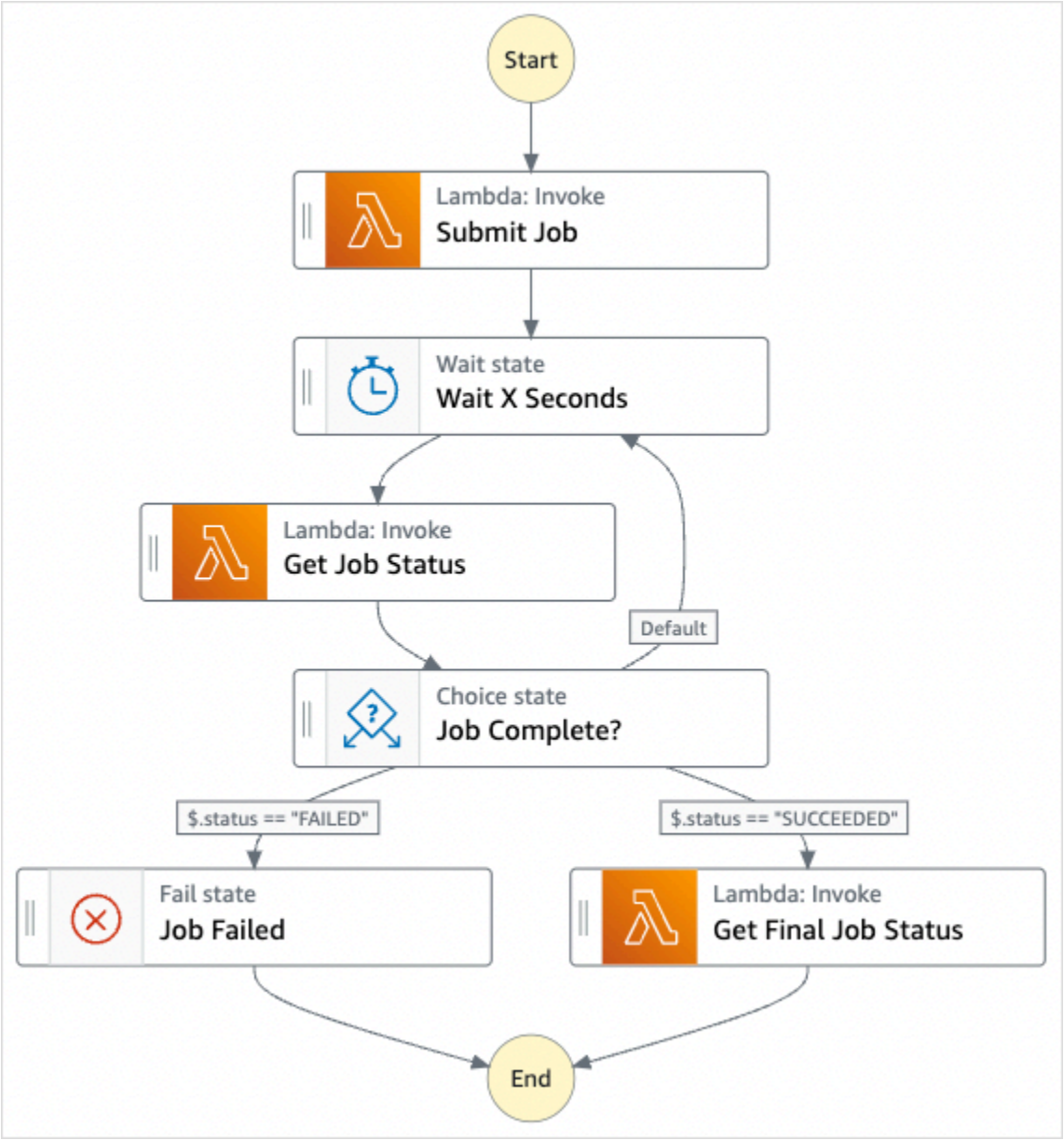
Fase 1: Creare la macchina a stati e fornire risorse

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`Job Poller`** nella casella di ricerca, quindi scegli Job Poller dai risultati di ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Questo progetto di esempio utilizza le seguenti risorse:

   - Tre funzioni Lambda per inviare un AWS Batch lavoro, ottenere lo stato corrente del AWS Batch lavoro inviato e lo stato di completamento del lavoro finale.
   - Un lavoro AWS Batch
   - Una macchina AWS Step Functions statale
   - Ruoli correlati AWS Identity and Access Management (IAM)

   L'immagine seguente mostra il grafico del flusso di lavoro per il progetto di esempio Job Poller:



5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo di workflow per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

     In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step

Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

> **Important**
>
> Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

- Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

  > **Tip**
  >
  > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

  Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

  La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Durante la distribuzione delle risorse, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

  Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

  > **Important**
  >
  > Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

Dopo aver fornito e distribuito tutte le risorse, viene visualizzata la finestra di dialogo Avvia esecuzione con un input di esempio simile al seguente.

```
{
    "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` impone allo stato `Wait` un loop ogni 60 secondi.

- Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

  1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

     > **Note**
     >
     > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

  2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

     Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

     > **Note**
     >
     > Se il progetto demo che hai distribuito contiene dati di input di esecuzione precompilati, usa quell'input per eseguire la macchina a stati.

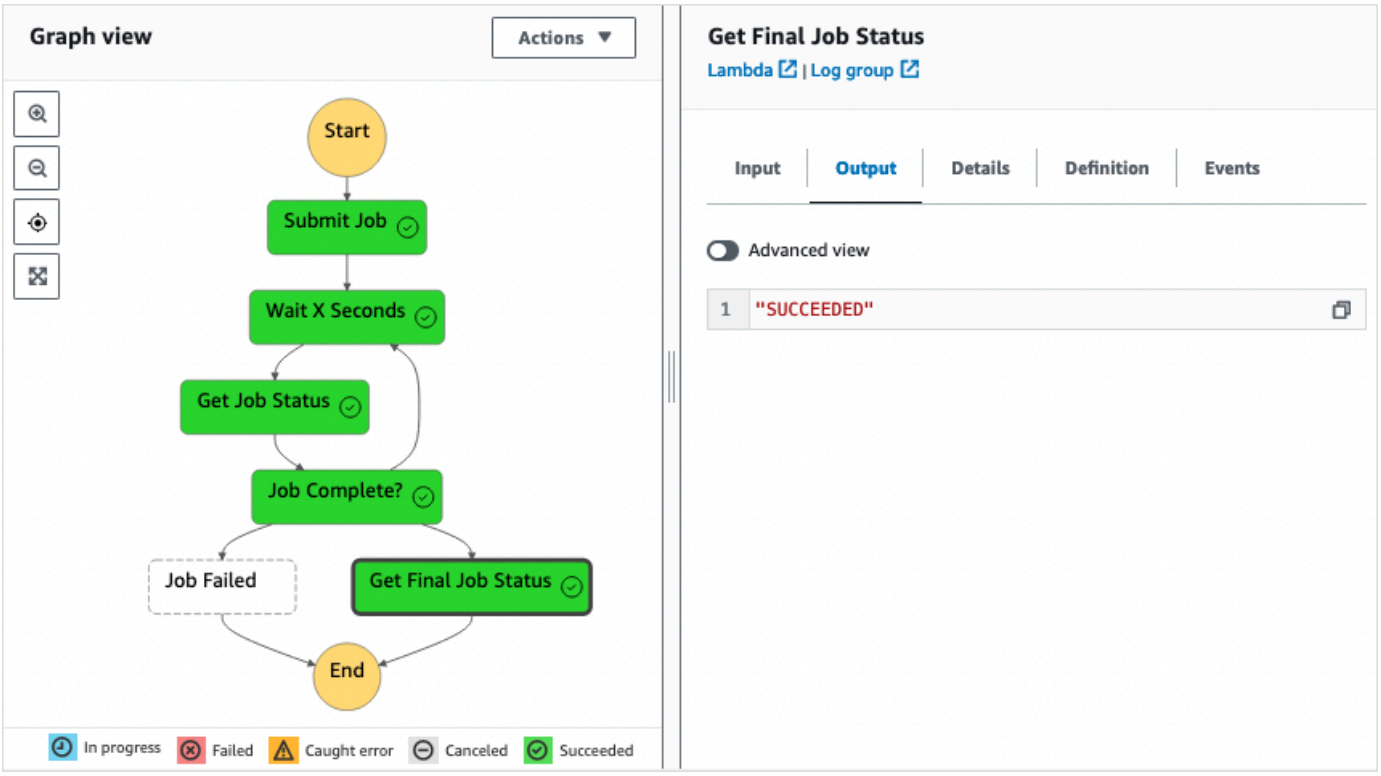
  3. Selezionare Start execution (Avvia esecuzione).

4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

   Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Ad esempio, per visualizzare il cambiamento dello stato del AWS Batch processo e i risultati ciclici dell'esecuzione, scegli la scheda Output.

L'immagine seguente mostra il grafico dello stato dell'esecuzione nella vista Grafico. Mostra anche l'output di esecuzione per la fase selezionata nella scheda Output.



Esempio di codice della macchina a stati

La macchina a stati di questo progetto di esempio si integra con l'invio AWS Lambda di un lavoro. AWS Batch Sfoglia questo esempio di macchina a stati per vedere come Step Functions controlla Lambda e. AWS Batch

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consultaUtilizzo AWS Step Functions con altri servizi.

```
{
    "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 delle attività (Lambda, Amazon SNS)

Il progetto prevede la creazione di un timer per attività. Implementa una macchina a AWS Step Functions stati che implementa `Wait` uno stato e utilizza una AWS Lambda funzione che invia una notifica Amazon Simple Notification Service (Amazon SNS). Il tipo di stato Attendi corrisponde all'attesa dell'esecuzione di un'unità di lavoro da parte di un trigger.

> **Note**
>
> Questo progetto di esempio implementa una AWS Lambda funzione per inviare una notifica Amazon Simple Notification Service (Amazon SNS). Puoi anche inviare una notifica Amazon SNS direttamente da Amazon States Language. Per informazioni, consulta Utilizzo AWS Step Functions con altri servizi.

Questo progetto di esempio crea la macchina a stati, una funzione Lambda e un argomento Amazon SNS e configura le AWS Identity and Access Management relative autorizzazioni (IAM). Per ulteriori informazioni sulle risorse create con il progetto Task Timer (Timer per attività), consulta:

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consulta. Utilizzo AWS Step Functions con altri servizi

- AWS CloudFormation Guida per l'utente
- Guida per sviluppatori di Amazon Simple Notification Service
- AWS Lambda Guida per gli sviluppatori
- Guida introduttiva a IAM

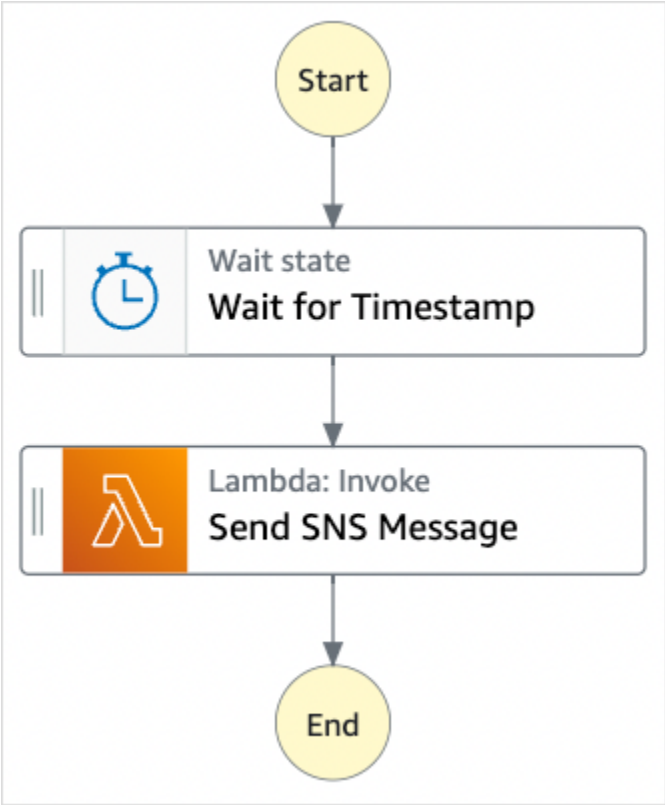
Fase 1: Creare la macchina a stati e fornire risorse

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`Task Timer`** nella casella di ricerca, quindi scegli Task Timer dai risultati della ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Questo progetto di esempio utilizza le seguenti risorse:

   - una funzione Lambda che invia una notifica Amazon SNS.
   - Una AWS Step Functions macchina a stati
   - Ruoli correlati AWS Identity and Access Management (IAM)

   L'immagine seguente mostra il grafico del flusso di lavoro per il progetto di esempio Task Timer:



5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo di workflow per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

     In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

     > **Important**
     >
     > Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

- Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

  > **Tip**
  >
  > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

  Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

  La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Durante la distribuzione delle risorse, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

  Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

  > **Important**
  >
  > Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

Dopo aver fornito e distribuito tutte le risorse, viene visualizzata la finestra di dialogo Avvia esecuzione con un input di esempio simile al seguente.

```
{
    "jobName": "my-job",{
  "topic": "arn:aws:sns:us-east-2:123456789012:StepFunctionsSample-TaskTimercc68840e-
c3d3-42a8-911e-821b7ce248e5-SNSTopic-44UjcFxzhACT",
  "message": "HelloWorld",
  "timer_seconds": 10
}
```

- Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

  1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

     > **Note**
     >
     > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

  2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

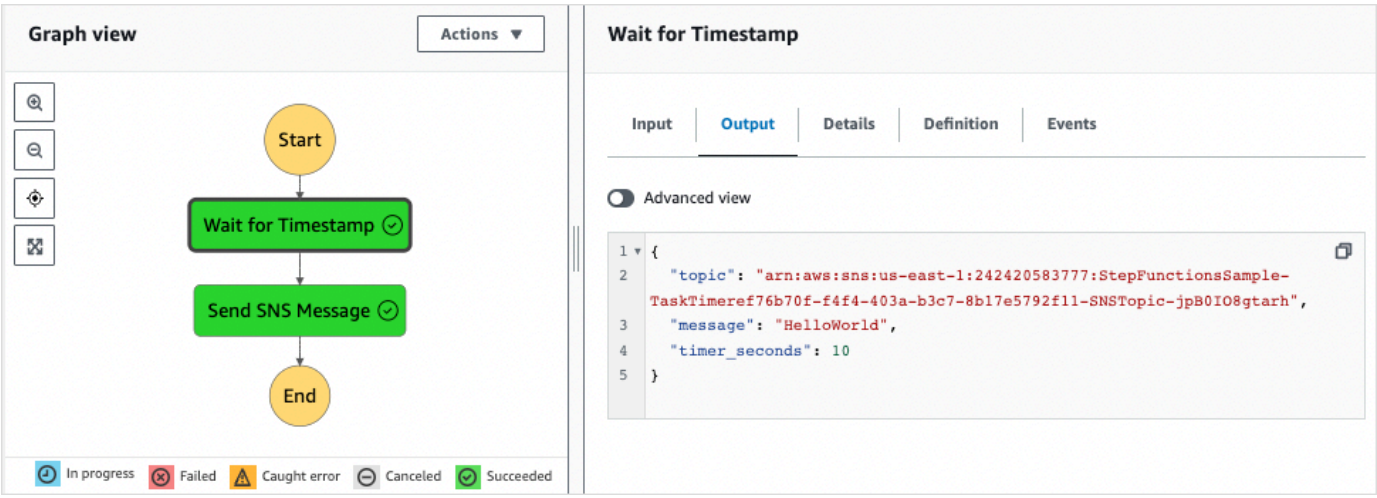
     Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

     > **Note**
     >
     > Se il progetto demo che hai distribuito contiene dati di input di esecuzione precompilati, usa quell'input per eseguire la macchina a stati.

  3. Selezionare Start execution (Avvia esecuzione).
  4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

     Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Ad esempio, l'immagine seguente mostra l'output del passaggio selezionato Wait for Timestamp. L'output di questo passaggio viene passato come input al passaggio Invia messaggio SNS.



Esempio di pattern di callback (Amazon SQS, Amazon SNS, Lambda)

Questo progetto di esempio dimostra come mettere in AWS Step Functions pausa un'attività e attendere che un processo esterno restituisca un token di attività generato all'avvio dell'attività.

Quando questo progetto di esempio viene distribuito e viene avviata l'esecuzione, si verificano i seguenti passaggi:

1. Step Functions passa un messaggio che include un token di attività a una coda Amazon Simple Queue Service (Amazon SQS).
2. Step Functions fa quindi una pausa, in attesa che quel token venga restituito.
3. La coda Amazon SQS attiva una AWS Lambda funzione che chiama `SendTaskSuccess`con lo stesso token di attività.
4. Quando viene ricevuto il token dell'attività, il flusso di lavoro continua.
5. L'`"Notify Success"`attività pubblica un messaggio Amazon Simple Notification Service (Amazon SNS) che comunica che la richiamata è stata ricevuta.

Per informazioni su come implementare il pattern di callback in Step Functions, vedereAttendere un callback con il token dell'attività.

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consultaUtilizzo AWS Step Functions con altri servizi.

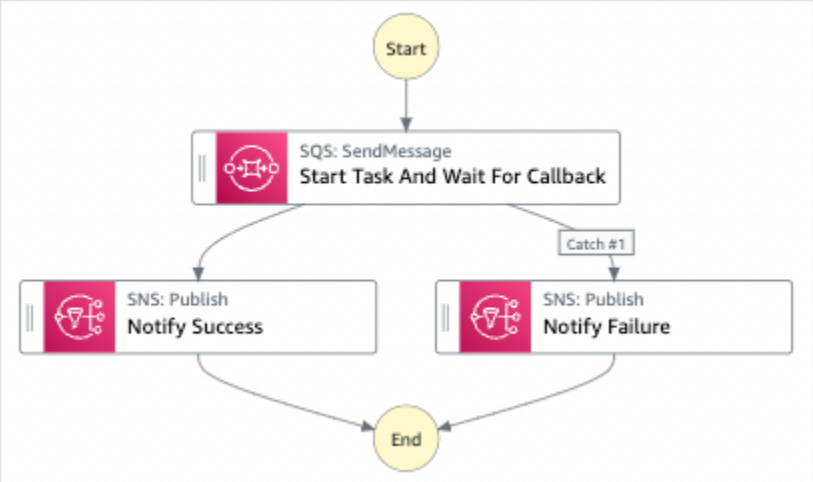
Fase 1: Creare la macchina a stati e fornire risorse

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`Callback pattern example`** nella casella di ricerca, quindi scegli Esempio di pattern di callback dai risultati della ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Questo progetto di esempio utilizza le seguenti risorse:

   - Una coda di messaggi Amazon SQS.
   - Una funzione Lambda che richiama l'azione dell'API Step Functions. SendTaskSuccess
   - Un argomento di Amazon SNS per notificare l'esito positivo o negativo di un'attività che indica se il flusso di lavoro può continuare o meno.
   - Una macchina a AWS Step Functions stati
   - Ruoli correlati AWS Identity and Access Management (IAM)

   L'immagine seguente mostra il grafico del flusso di lavoro per il progetto di esempio del pattern Callback:



5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

- Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

  In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

  > **Important**
  >
  > Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

- Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

  > **Tip**
  >
  > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

  Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

  La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Mentre le risorse vengono distribuite, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

  Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

> **Important**
>
> Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.
2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

    1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

        > **Note**
        >
        > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

    2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

        Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

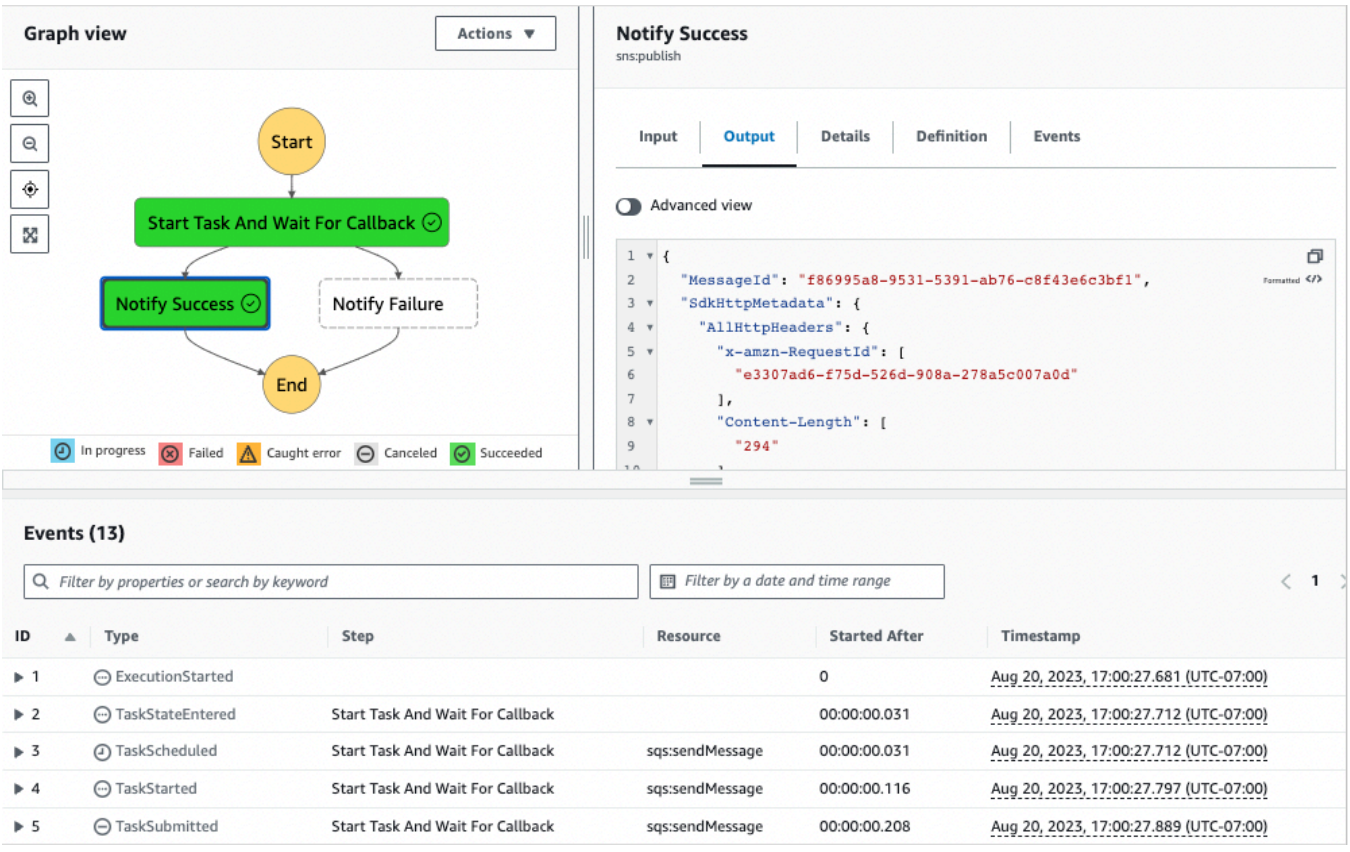
        > **Note**
        >
        > Se il progetto demo che hai distribuito contiene dati di input di esecuzione precompilati, usa quell'input per eseguire la macchina a stati.

    3. Selezionare Start execution (Avvia esecuzione).
    4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile

esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vedere Pagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Ad esempio, per verificare in che modo Step Functions è progredita nel flusso di lavoro e ha ricevuto una richiamata da Amazon SQS, esamina le voci nella tabella Eventi. L'immagine seguente mostra l'output dell'esecuzione per la fase Notify Success. Mostra anche i primi cinque eventi della cronologia degli eventi di esecuzione. Espandi ogni evento per visualizzare ulteriori dettagli su quell'evento.



Esempio di callback Lambda

Per vedere come interagiscono i componenti di questo progetto di esempio, guarda le risorse che sono state distribuite nel tuo account. AWS Ad esempio, ecco la funzione Lambda che chiama Step Functions con il token task.

```
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);
        });
    }
};
```

Gestisci un lavoro in Amazon EMR

Questo progetto di esempio illustra Amazon EMR AWS Step Functions e l'integrazione.

Mostra come creare un cluster Amazon EMR, aggiungere più passaggi ed eseguirli, quindi terminare il cluster.

> **Important**
>
> Amazon EMR non prevede un livello di prezzo gratuito. L'esecuzione del progetto di esempio comporta costi. Puoi trovare informazioni sui prezzi nella pagina dei prezzi di Amazon EMR. La disponibilità dell'integrazione del servizio Amazon EMR è soggetta alla disponibilità delle API di Amazon EMR. Per questo motivo, questo progetto di esempio potrebbe non funzionare correttamente in alcune regioni. AWS Consulta la documentazione di Amazon EMR per le limitazioni in regioni speciali.

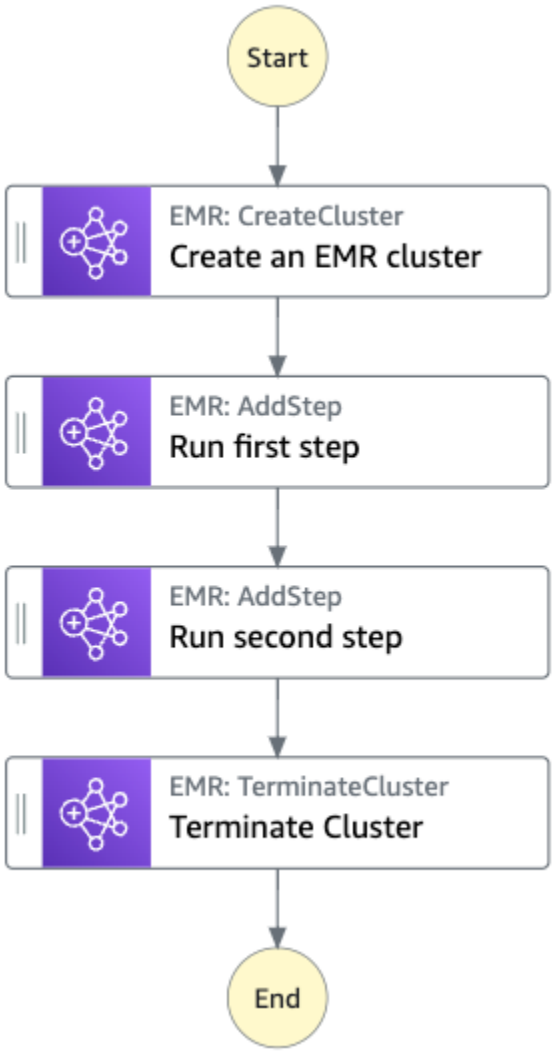
Fase 1: creazione della State Machine e fornitura di risorse

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`Manage an EMR job`** nella casella di ricerca, quindi scegli Gestisci un lavoro EMR dai risultati di ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Questo progetto di esempio utilizza le seguenti risorse:

   - Un bucket Amazon S3
   - Un cluster Amazon EMR
   - Una macchina a AWS Step Functions stati
   - Ruoli correlati AWS Identity and Access Management (IAM)

   L'immagine seguente mostra il grafico del flusso di lavoro per il progetto di esempio Manage an EMR Job:



5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

     In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

> **Important**
>
> Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

- Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

  > **Tip**
  >
  > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

  Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

  La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Mentre le risorse vengono distribuite, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

  Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

  > **Important**
  >
  > Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.
2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

   > **Note**
   >
   > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

   Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

   > **Note**
   >
   > Se il progetto demo che hai distribuito contiene dati di input di esecuzione precompilati, usa quell'input per eseguire la macchina a stati.

3. Selezionare Start execution (Avvia esecuzione).
4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

   Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Esempio di codice della macchina a stati

La macchina a stati di questo progetto di esempio si integra con Amazon EMR passando i parametri direttamente a tali risorse. Sfoglia questo esempio di macchina a stati per vedere come Step

Functions utilizza una macchina a stati per chiamare l'attività Amazon EMR in modo sincrono, attende che l'operazione abbia successo o meno e termina il cluster.

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consulta. Utilizzo AWS Step Functions con altri servizi

```
{
  "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
    }
  }
}
```

Esempio IAM

Questo esempio di policy AWS Identity and Access Management (IAM) generato dal progetto di esempio include il minimo privilegio necessario per eseguire la macchina a stati e le risorse correlate. È consigliabile includere solo le autorizzazioni necessarie nelle policy IAM.

```
{
    "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"
            ]
        }
    ]
}
```

La policy seguente garantisce che `addStep` disponga di autorizzazioni sufficienti.

```
{
    "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"
            ]
        }
    ]
}
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Esegui un EMR Serverless lavoro

Questo progetto di esempio dimostra come creare e avviare un'applicazione. EMR Serverless Questo progetto mostra anche come eseguire più lavori all'interno di quell'applicazione.

Questo progetto di esempio crea la macchina a stati, le AWS risorse di supporto e configura le relative autorizzazioni IAM. Esplora questo progetto di esempio per imparare a eseguire EMR Serverless lavori utilizzando macchine a Step Functions stati o usalo come punto di partenza per i tuoi progetti.

> **Important**
>
> EMR Serverless non dispone di un piano tariffario gratuito. L'esecuzione del progetto di esempio comporta costi. Per informazioni dettagliate sui prezzi, visita la pagina dei prezzi di Amazon EMR Serverless.
> Inoltre, la disponibilità dell'integrazione dei EMR Serverless servizi è soggetta alla disponibilità delle EMR Serverless API. Per questo motivo, questo progetto di esempio potrebbe non funzionare correttamente o essere disponibile in alcuni Regioni AWS. Per informazioni sulla disponibilità di in, consulta l'argomento Altre considerazioni. EMR Serverless Regioni AWS

Modello AWS CloudFormation e risorse aggiuntive

Si utilizza un CloudFormation modello per distribuire questo progetto di esempio. Questo modello crea le seguenti risorse nel tuo Account AWS:

- Una macchina a Step Functions stati.
- Ruolo di esecuzione per la macchina a stati. Questo ruolo concede le autorizzazioni necessarie alla macchina a stati per accedere ad altre Servizi AWS risorse come l'azione EMR Serverless CreateApplication.
- Ruolo di esecuzione del lavoro perEMR Serverless. Questo ruolo concede le autorizzazioni che un EMR Serverless job run può assumere quando chiama altri servizi per conto dell'utente.

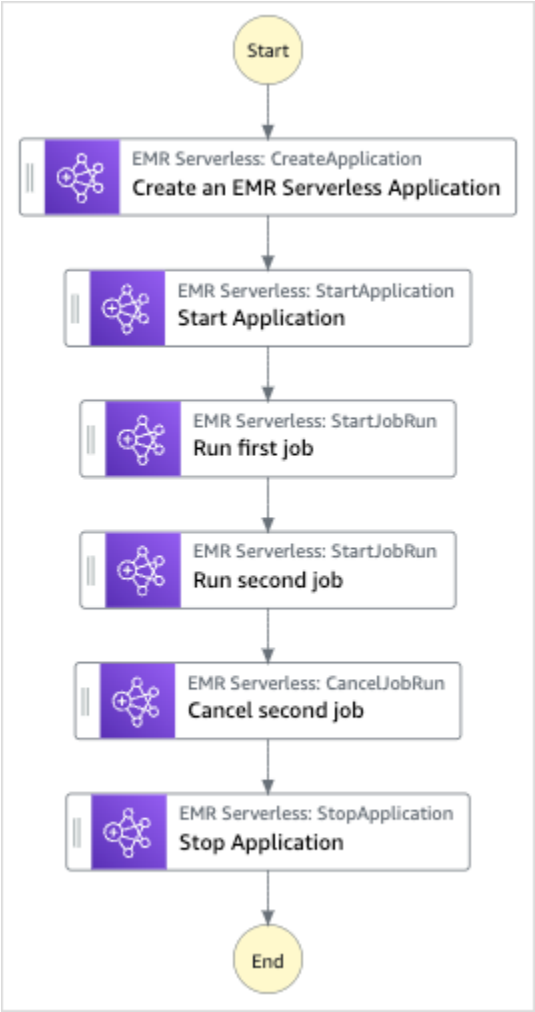
Fase 1: Creare la macchina a stati e fornire risorse

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`EMR Serverless`** nella casella di ricerca, quindi scegli Esegui un EMR Serverless lavoro dai risultati della ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Questo progetto di esempio utilizza le seguenti risorse:

   - Una macchina a stati Step Functions
   - Ruoli AWS Identity and Access Management (IAM) correlati

   L'immagine seguente mostra il grafico del flusso di lavoro per il progetto di esempio Run an EMR Serverless job:



5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

     In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

> **Important**
>
> Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

- Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

  > **Tip**
  >
  > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

  Quando sei pronto, scegli Implementa ed esegui per distribuire il progetto di esempio e creare le risorse.

  La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Mentre le risorse vengono distribuite, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

  Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

  > **Important**
  >
  > Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.
2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

   > **Note**
   >
   > Step Functions consente di creare nomi per macchine a stati, esecuzioni, attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

   Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

3. Selezionare Start execution (Avvia esecuzione).

4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

   Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Avvio di un workflow all'interno di un workflow (Step Functions, Lambda)

Questo progetto di esempio dimostra come utilizzare una macchina a AWS Step Functions stati per avviare altre esecuzioni di macchine a stati. Per informazioni sull'avvio delle esecuzioni di macchine a stati da un'altra macchina a stati, vedere. Avvio delle esecuzioni di flusso di lavoro da uno stato di task

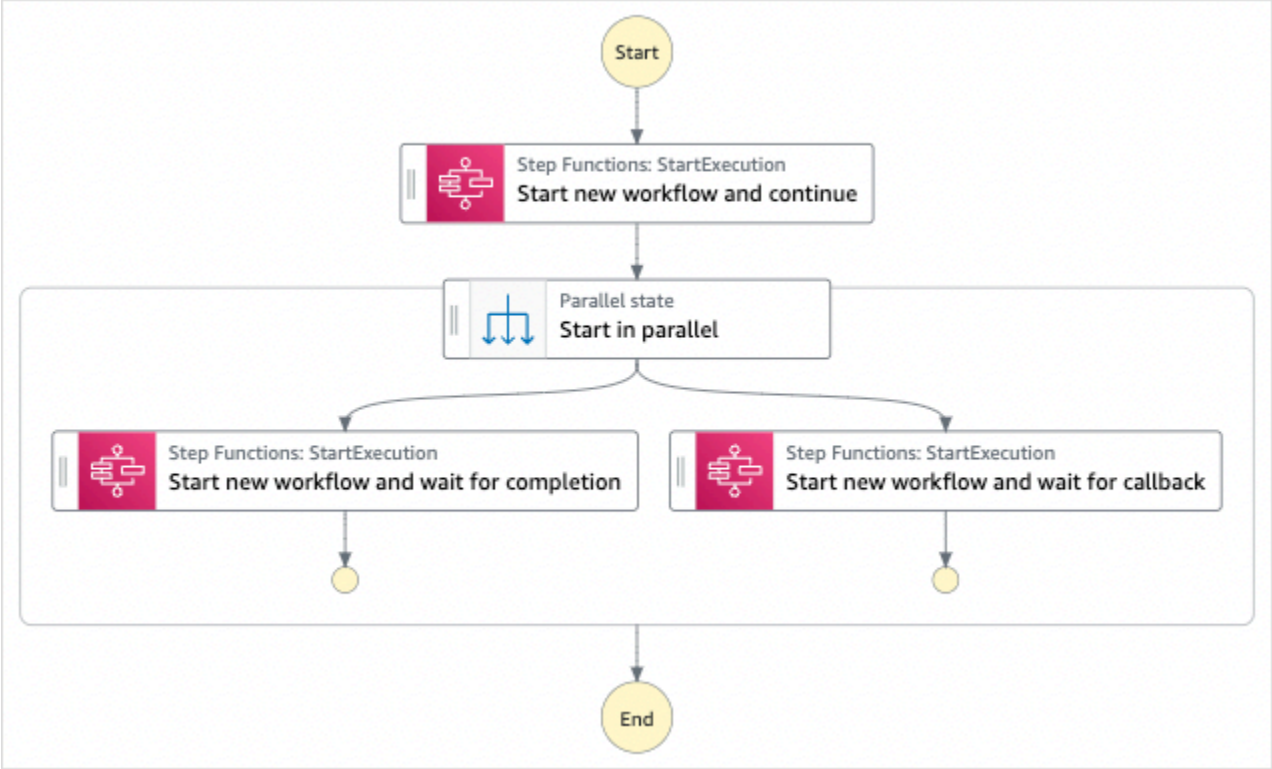
Fase 1: Creare la macchina a stati e fornire risorse

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`Start a workflow within a workflow`** nella casella di ricerca, quindi scegli Avvia un flusso di lavoro all'interno di un flusso di lavoro dai risultati della ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Questo progetto di esempio utilizza le seguenti risorse:

   - Una macchina a stati aggiuntiva. L'esecuzione di questa macchina a stati viene avviata dalla macchina a stati in esecuzione.
   - Una funzione Lambda di callback. Questa funzione viene utilizzata nella macchina a stati aggiuntiva per implementare il meccanismo di callback.
   - Una macchina a AWS Step Functions stati
   - Ruoli correlati AWS Identity and Access Management (IAM)

   L'immagine seguente mostra il grafico del flusso di lavoro per il progetto di esempio Avvia un flusso di lavoro all'interno di un progetto di esempio:



5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

     In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

     > **Important**
     >
     > Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

- Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

  > **Tip**
  >
  > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

  Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

  La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Mentre le risorse vengono distribuite, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

  Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

  > **Important**
  >
  > Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.
2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

   1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

> **Note**
>
> Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

   Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

   > **Note**
   >
   > Se il progetto demo che hai distribuito contiene dati di input di esecuzione precompilati, usa quell'input per eseguire la macchina a stati.

3. Selezionare Start execution (Avvia esecuzione).
4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

   Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Esempio di codice della macchina a stati

La macchina a stati di questo progetto di esempio integra un'altra macchina a stati e AWS Lambda passa i parametri direttamente a tali risorse.

Sfoglia questo esempio di macchina a stati per vedere come Step Functions chiama l'azione `StartExecution` API per l'altra macchina a stati. Avvia due istanze dell'altra macchina a stati in

parallelo: una utilizzando il modello Esegui un processo (.sync) e una utilizzando il modello Attendere un callback con il token dell'attività.

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consultaUtilizzo AWS Step Functions con altri servizi.

```
{
  "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
            }
          }
        }
      ]
    }
  }
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Elabora dinamicamente i dati con uno stato della mappa

Questo progetto di esempio mostra il parallelismo dinamico utilizzando uno stato Eseguire la mappatura.

In questo progetto, Step Functions utilizza una AWS Lambda funzione per estrarre i messaggi da una coda Amazon SQS e passare un array JSON di tali messaggi a uno stato. `Map` Per ogni messaggio in coda, la macchina a stati scrive il messaggio su DynamoDB, richiama l'altra funzione Lambda per rimuovere il messaggio da Amazon SQS e quindi pubblica il messaggio nell'argomento Amazon SNS.

Per ulteriori informazioni sugli `Map` stati e sulle integrazioni dei servizi Step Functions, consulta quanto segue:

- Eseguire la mappatura
- Utilizzo AWS Step Functions con altri servizi

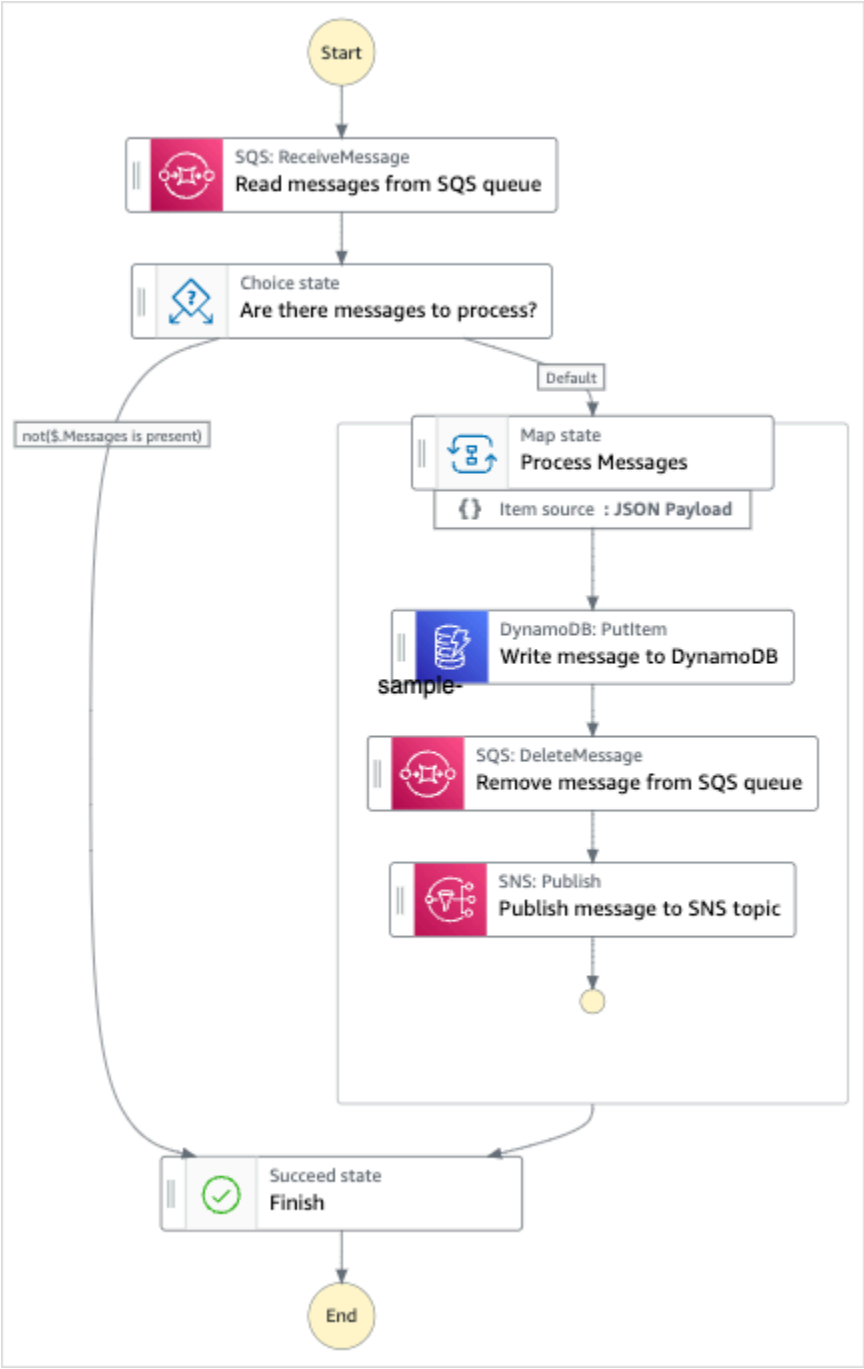
Fase 1: Creare la macchina a stati e fornire risorse

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`Dynamically process data with a Map state`** nella casella di ricerca, quindi scegli Elabora dinamicamente i dati con uno stato della mappa dai risultati della ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Questo progetto di esempio utilizza le seguenti risorse:

   - Una coda Amazon SQS da cui lo stato Map legge e rimuove i messaggi in modo iterativo.
   - Una tabella DynamoDB in cui lo stato Map scrive i messaggi in modo iterativo.
   - Un argomento di Amazon SNS in cui Step Functions pubblica i messaggi letti dalla coda Amazon SQS.
   - Due funzioni AWS Lambda
   - Una macchina a AWS Step Functions stati
   - Ruoli correlati AWS Identity and Access Management (IAM)

   L'immagine seguente mostra il grafico del flusso di lavoro per il progetto di esempio di elaborazione dinamica dei dati con un progetto di esempio dello stato della mappa:



5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

> **Important**
>
> Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

- Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

  > **Tip**
  >
  > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

  Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

  La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Mentre le risorse vengono distribuite, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

  Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

  > **Important**
  >
  > Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Dopo aver distribuito le risorse del progetto di esempio, devi aggiungere elementi alla coda di Amazon SQS e iscriverti all'argomento Amazon SNS prima di eseguire la macchina a stati.

Fase 2: Abbonarsi all'argomento Amazon SNS

1. Apri la console Amazon SNS.
2. Scegliere Topics (Argomenti) e scegliere l'argomento creato dal progetto di esempio dello stato `Map`.

   Il nome sarà simile a MapSampleProj-SNSTopic-1CQO4HQ3IR1kN.
3. Scegli Crea sottoscrizione.

   Viene visualizzata la pagina Create subscription (Crea sottoscrizione) con l'elenco Topic ARN (ARN argomento) per l'argomento.
4. Per Protocol (Protocollo), scegliere Email.
5. In Endpoint immettere un indirizzo e-mail per effettuare la sottoscrizione all'argomento.
6. Scegli Crea sottoscrizione.

   > **Note**
   >
   > Devi confermare la sottoscrizione nella tua e-mail prima che sia attiva.
7. Aprire l'e-mail Subscription Confirmation (Conferma sottoscrizione) nell'account correlato e aprire l'URL Confirm subscription (Conferma sottoscrizione).

   Viene visualizzata la pagina Subscription confirmed! (Sottoscrizione confermata!).

Fase 3: Aggiungere messaggi alla coda Amazon SQS

1. Apri la console Amazon SQS.
2. Scegliere la coda creata dal progetto di esempio dello stato `Map`.

   Il nome sarà simile a -sqsqueue-1Udic9vzdorn7MapSampleProj.
3. Scegli Invia e ricevi messaggi.
4. Nella pagina Invia e ricevi messaggi, inserisci un messaggio e scegli Invia messaggio.
5. Inserisci un altro messaggio e scegli Invia messaggio. Continua a inserire altri messaggi finché non ne avrai diversi nella coda di Amazon SQS.

Fase 4: Esegui la macchina a stati

> **Note**
>
> Le code in Amazon SNS alla fine sono coerenti. Per ottenere risultati ottimali, attendere alcuni minuti tra il popolamento della coda e l'avvio di un'esecuzione della macchina a stati.

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.
2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

    1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

        > **Note**
        >
        > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

    2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

        Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

        > **Note**
        >
        > Se il progetto demo che hai distribuito contiene dati di input di esecuzione precompilati, usa quell'input per eseguire la macchina a stati.

    3. Selezionare Start execution (Avvia esecuzione).
    4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vedere Pagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Esempio di codice di macchina a stati

La macchina a stati di questo progetto di esempio si integra con Amazon SQS, Amazon SNS e Lambda passando i parametri direttamente a tali risorse.

Sfoglia questo esempio di macchina a stati per vedere come Step Functions controlla Lambda, DynamoDB, Amazon SNS connettendosi all'Amazon Resource Name (ARN) sul campo e passando all'API del `Resource` servizio. `Parameters`

Per ulteriori informazioni su come controllare altri servizi, AWS Step Functions consulta. AWS Utilizzo AWS Step Functions con altri servizi

```
{
  "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"
    }
  }
}
```

Esempio IAM

Questo esempio di policy AWS Identity and Access Management (IAM) generato dal progetto di esempio include il minimo privilegio necessario per eseguire la macchina a stati e le risorse correlate. Ti consigliamo di includere solo le autorizzazioni necessarie nelle tue policy IAM.

```
{
    "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"
        }
    ]
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Elabora un file CSV con Distributed Map

Questo progetto di esempio dimostra come utilizzare lo stato Distributed Map per iterare oltre 10.000 righe di un file CSV generato utilizzando una funzione. Lambda Il file CSV contiene le informazioni di spedizione degli ordini dei clienti ed è archiviato in un bucket Amazon S3. La mappa distribuita esegue iterazioni su un batch di 10 righe nel file CSV per l'analisi dei dati.

La mappa distribuita contiene una Lambda funzione per rilevare eventuali ordini in ritardo. La mappa distribuita contiene anche una mappa in linea per elaborare gli ordini in ritardo in un batch e restituisce questi ordini in ritardo in un array. Per ogni ordine ritardato, la Inline Map invia un messaggio a una coda. Amazon SQS Infine, questo progetto di esempio memorizza i risultati di Map Run in un altro bucket Amazon S3 del tuo. Account AWS

Con Distributed Map, puoi eseguire fino a 10.000 esecuzioni parallele di workflow secondari alla volta. In questo progetto di esempio, la concorrenza massima di Distributed Map è impostata su 1000, il che la limita a 1000 esecuzioni di workflow secondarie parallele.

Questo progetto di esempio crea la macchina a stati, le AWS risorse di supporto e configura le relative autorizzazioni IAM. Esplora questo progetto di esempio per scoprire come utilizzare la Distributed Map per orchestrare carichi di lavoro paralleli su larga scala o usala come punto di partenza per i tuoi progetti.

AWS CloudFormation modello e risorse aggiuntive

Si utilizza un CloudFormation modello per distribuire questo progetto di esempio. Questo modello crea le seguenti risorse nel tuo Account AWS:

- Una macchina a stati Step Functions.
- Ruolo di esecuzione per la macchina a stati. Questo ruolo concede le autorizzazioni necessarie alla macchina a stati per accedere ad altre Servizi AWS risorse come l'azione Invoke della funzione Lambda.
- Una funzione Lambda denominata `CSVGeneratorFunction` che genera un file CSV che contiene i dettagli dell'ordine del cliente.
- Ruolo di esecuzione per la funzione Lambda del generatore CSV. Questo ruolo concede alla funzione il permesso di accedere ad altri. Servizi AWS
- Un bucket di input Amazon S3 per archiviare il file CSV generato.
- Una funzione Lambda per il rilevamento ritardato degli ordini che analizza i dati del file CSV e rileva eventuali ordini in ritardo.
- Ruolo di esecuzione per la funzione Lambda con ordine ritardato. Questo ruolo concede alla funzione il permesso di accedere ad altri. Servizi AWS
- Un bucket di output Amazon S3 per archiviare i risultati dell'analisi degli ordini dei clienti.
- Una coda Amazon SQS a cui Step Functions invia messaggi per ogni ordine in ritardo. Questi messaggi contengono gli ID dei clienti e i loro ordini.
- Un gruppo di CloudWatch log che memorizza le informazioni relative alla cronologia di esecuzione della macchina a stati.

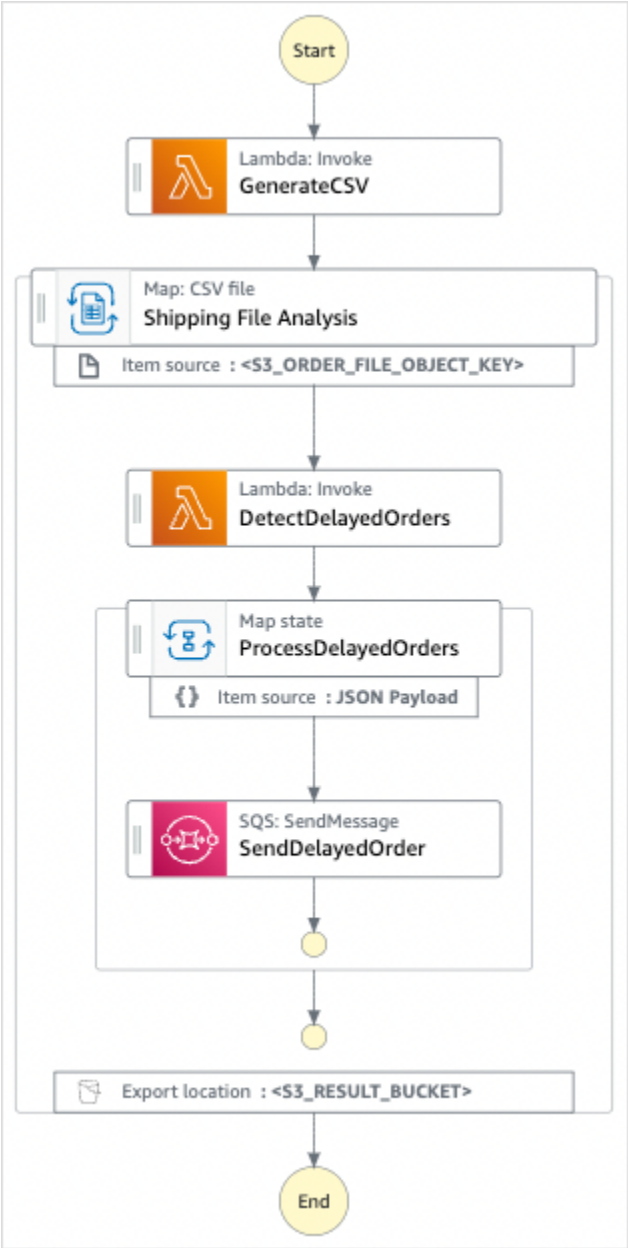
> **Important**
>
> Per ogni servizio sono previsti costi standard.

Fase 1: Creare la macchina a stati e fornire risorse

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`Distributed Map to process a CSV file in S3`** nella casella di ricerca, quindi scegli Mappa distribuita per elaborare un file CSV in S3 dai risultati della ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Per informazioni sulle risorse che verranno create per questo progetto di esempio, consultaAWS CloudFormation modello e risorse aggiuntive.

   L'immagine seguente mostra il grafico del flusso di lavoro per la Distributed Map per elaborare un file CSV nel progetto di esempio S3:



5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

     In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione

Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

> **Important**
>
> Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

- Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

  > **Tip**
  >
  > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

  Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

  La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Durante la distribuzione delle risorse, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

  Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

  > **Important**
  >
  > Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

Dopo aver effettuato il provisioning e la distribuzione di tutte le risorse, è possibile eseguire la macchina a stati.

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.
2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

   a. (Facoltativo) Immettete i valori di input in formato JSON per eseguire il progetto di esempio.

      Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

      > **Note**
      >
      > Se il progetto demo che hai distribuito contiene dati di input di esecuzione precompilati, usa quell'input per eseguire la macchina a stati.

   b. Selezionare Start execution (Avvia esecuzione).

   c. (Facoltativo) La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

      Al termine dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione.

      - Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.
      - Per ulteriori informazioni sulla visualizzazione dell'esecuzione dello stato di una mappa distribuita nella console, vedereEsame di Map Run.

   d. (Facoltativo) Esamina i risultati dell'esecuzione esportati nel bucket Amazon S3. Questi risultati includono dati, come input e output di esecuzione, ARN e stato di esecuzione. Per ulteriori informazioni, consulta ResultWriter.

Elabora i dati in un bucket Amazon S3 con Distributed Map

Questo progetto di esempio dimostra come è possibile utilizzare lo stato della mappa distribuita per elaborare dati su larga scala, ad esempio analizzare dati meteorologici storici e identificare la stazione meteorologica con la temperatura media più alta del pianeta ogni mese. I dati meteorologici vengono registrati in oltre 12.000 file CSV, che a loro volta vengono archiviati in un bucket Amazon S3.

Questo progetto di esempio include due stati della mappa distribuiti denominati Distributed S3 copy NOA Data e ProcessNoAAData. Copia distribuita di S3 I dati NOA eseguono iterazioni sui file CSV in un bucket Amazon S3 pubblico denominato e noaa-gsod-pdsli copia in un bucket Amazon S3 del tuo. Account AWS ProcessNoAAData esegue un'iterazione sui file copiati e include una funzione Lambda che esegue l'analisi della temperatura.

Il progetto di esempio verifica innanzitutto il contenuto del bucket Amazon S3 con una chiamata all'azione API ListObjectsV2. In base al numero di chiavi restituite in risposta a questa chiamata, il progetto di esempio prende una delle seguenti decisioni:

- Se il conteggio delle chiavi è maggiore o uguale a 1, il progetto passa allo stato processNoAAData. Questo stato della mappa distribuita include una Lambda funzione denominata TemperatureFunctionche trova la stazione meteorologica con la temperatura media più alta ogni mese. Questa funzione restituisce un dizionario con `year-month` come chiave e un dizionario che contiene informazioni sulla stazione meteorologica come valore.
- Se il numero di chiavi restituito non supera 1, lo stato dei dati NOA di Distributed S3 copy elenca tutti gli oggetti dal bucket pubblico noaa-gsod-pdse copia iterativamente i singoli oggetti in un altro bucket dell'account in batch da 100. Una mappa in linea esegue la copia iterativa degli oggetti.

  Dopo aver copiato tutti gli oggetti, il progetto passa allo stato processNoAAData per l'elaborazione dei dati meteorologici.

Il progetto di esempio passa infine a una Lambda funzione di riduzione che esegue un'aggregazione finale dei risultati restituiti dalla funzione e scrive i risultati in una tabella. TemperatureFunctionAmazon DynamoDB

Con Distributed Map, puoi eseguire fino a 10.000 esecuzioni parallele di workflow secondari alla volta. In questo progetto di esempio, la concorrenza massima di ProcessNoAAData Distributed Map è impostata su 3000, il che la limita a 3000 esecuzioni parallele di flussi di lavoro secondari.

Questo progetto di esempio crea la macchina a stati, le AWS risorse di supporto e configura le relative autorizzazioni IAM. Esplora questo progetto di esempio per scoprire come utilizzare la Distributed Map per orchestrare carichi di lavoro paralleli su larga scala o usala come punto di partenza per i tuoi progetti.

> **Important**
>
> Questo progetto di esempio è disponibile solo nella regione Stati Uniti orientali (Virginia settentrionale).

AWS CloudFormation modello e risorse aggiuntive

Si utilizza un CloudFormation modello per distribuire questo progetto di esempio. Questo modello crea le seguenti risorse nel tuo Account AWS:

- Una macchina a stati Step Functions.
- Ruolo di esecuzione per la macchina a stati. Questo ruolo concede le autorizzazioni necessarie alla macchina a stati per accedere ad altre Servizi AWS risorse come l'azione Invoke della funzione Lambda.
- Un bucket Amazon S3 denominato. `N0AADataBucket` Questo bucket contiene i file CSV con i dati meteorologici.
- Una funzione Lambda denominata `ReducerFunction` che esegue un'aggregazione finale dei dati meteorologici e scrive i risultati in una tabella Amazon DynamoDB.
- Ruolo di esecuzione per la funzione Lambda del riduttore. Questo ruolo concede alla funzione il permesso di accedere ad altri. Servizi AWS
- Un bucket di output Amazon S3 denominato `ResultsBucket` per archiviare i risultati dell'analisi meteorologica.
- Una tabella DynamoDB `ResultsDynamoDBTable` denominata che contiene i risultati restituiti da. `ReducerFunction`
- Una funzione Lambda denominata `TemperatureFunction` che rileva la temperatura media mensile più alta.
- Ruolo di esecuzione per la funzione Lambda. Questo ruolo concede alla funzione il permesso di accedere ad altri. Servizi AWS
- Un gruppo di CloudWatch log che memorizza le informazioni relative alla cronologia di esecuzione della macchina a stati.

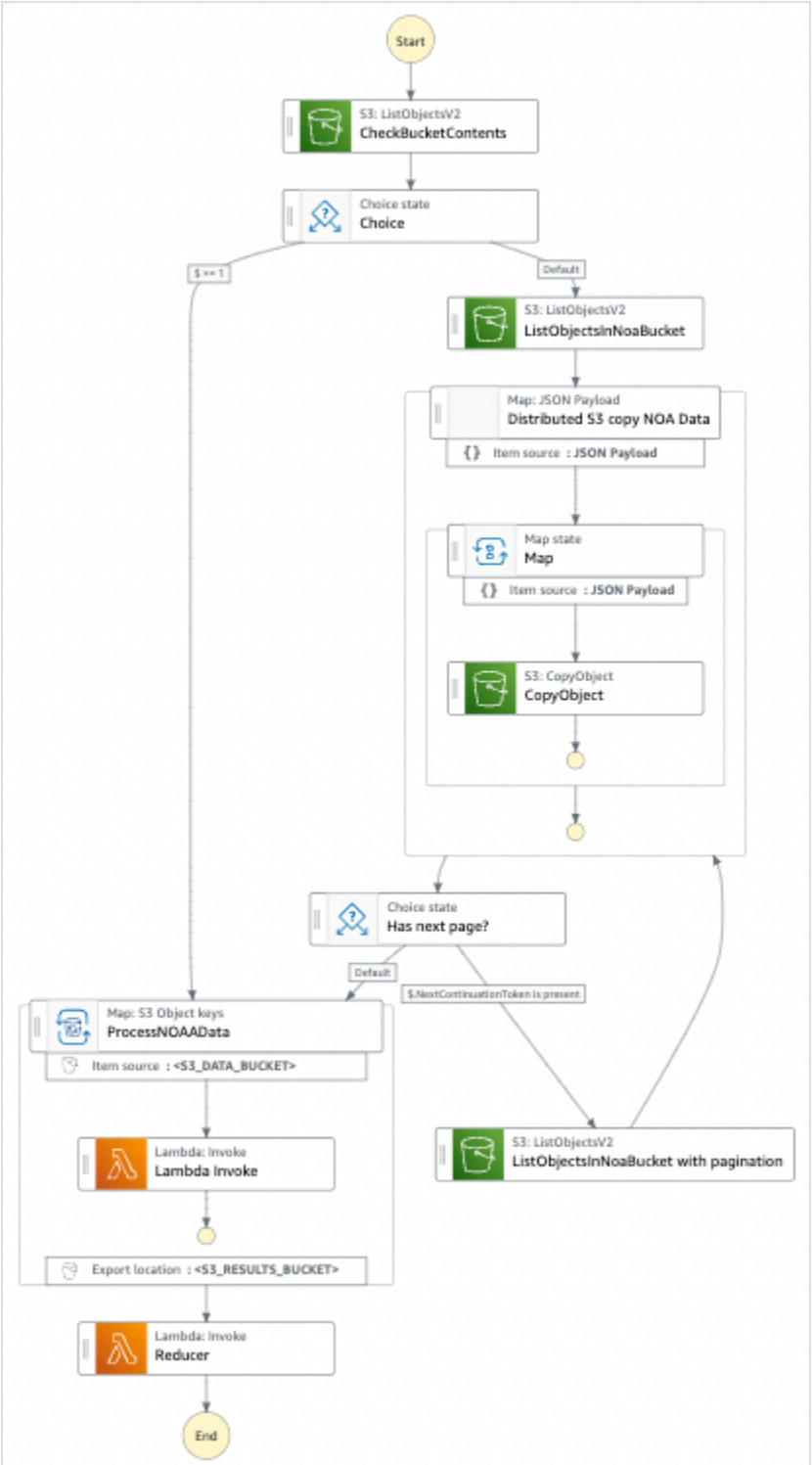
> **Important**
>
> Per ogni servizio sono previsti costi standard.

Fase 1: Creare la macchina a stati e fornire risorse

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`Distributed Map to process files in S3`** nella casella di ricerca, quindi scegli Mappa distribuita per elaborare i file in S3 dai risultati di ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Per informazioni sulle risorse che verranno create per questo progetto di esempio, consultaAWS CloudFormation modello e risorse aggiuntive.

   L'immagine seguente mostra il grafico del flusso di lavoro per la Distributed Map per elaborare i file nel progetto di esempio S3:



5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

- Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

  In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

  > **Important**
  >
  > Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

- Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

  > **Tip**
  >
  > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

  Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

  La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Mentre le risorse vengono distribuite, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

  Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

> **Important**
>
> Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

Dopo aver effettuato il provisioning e la distribuzione di tutte le risorse, è possibile eseguire la macchina a stati.

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.
2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

   a. (Facoltativo) Immettete i valori di input in formato JSON per eseguire il progetto di esempio.

      Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

      > **Note**
      >
      > Se il progetto demo che hai distribuito contiene dati di input di esecuzione precompilati, usa quell'input per eseguire la macchina a stati.

   b. Selezionare Start execution (Avvia esecuzione).

   c. (Facoltativo) La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

      Al termine dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione.

      - Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vedere Pagina dei dettagli di esecuzione: panoramica dell'interfaccia.

- Per ulteriori informazioni sulla visualizzazione dell'esecuzione di uno stato di Distributed Map nella console, vedereEsame di Map Run.

d. (Facoltativo) Esamina i risultati dell'esecuzione esportati nel bucket Amazon S3. Questi risultati includono dati, come input e output di esecuzione, ARN e stato di esecuzione. Per ulteriori informazioni, consulta ResultWriter.

Training su un modello di Machine Learning

Questo progetto di esempio dimostra come utilizzare SageMaker e AWS Step Functions addestrare un modello di apprendimento automatico e come trasformare in batch un set di dati di test.

In questo progetto, Step Functions utilizza una funzione Lambda per seminare un bucket Amazon S3 con un set di dati di test. Quindi addestra un modello di machine learning ed esegue una trasformazione in batch, utilizzando l'integrazione del servizio. SageMaker

Per ulteriori informazioni sulle SageMaker integrazioni dei servizi Step Functions, consulta quanto segue:

- Utilizzo AWS Step Functions con altri servizi
- Gestisci SageMaker con Step Functions

> **Note**
>
> Questo progetto di esempio potrebbe comportare costi.
> Per i nuovi AWS utenti, è disponibile un livello di utilizzo gratuito. In questo piano, i servizi sono gratuiti al di sotto di un determinato livello di utilizzo. Per ulteriori informazioni sui AWS costi e sul piano gratuito, consulta la sezione SageMaker Prezzi.

Fase 1: Creazione della macchina a stati e fornitura delle risorse

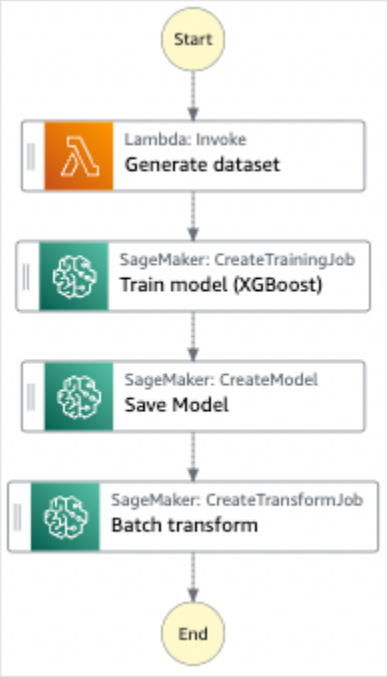
1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`Train a machine learning model`** nella casella di ricerca, quindi scegli Addestra un modello di machine learning dai risultati di ricerca restituiti.
3. Seleziona Successivo per continuare.

4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Questo progetto di esempio utilizza le seguenti risorse:

   - Una funzione AWS Lambda
   - Un bucket Amazon Simple Storage Service (Amazon S3).
   - Una macchina AWS Step Functions statale
   - Ruoli correlati AWS Identity and Access Management (IAM)

   L'immagine seguente mostra il grafico del flusso di lavoro per il progetto di esempio Train a machine learning model:



5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

> **Important**
>
> Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

- Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

  > **Tip**
  >
  > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

  Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

  La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Mentre le risorse vengono distribuite, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

  Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

  > **Important**
  >
  > Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.
2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

   1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

      > **Note**
      >
      > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

   2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

      Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

      > **Note**
      >
      > Se il progetto demo che hai distribuito contiene dati di input di esecuzione precompilati, usa quell'input per eseguire la macchina a stati.

   3. Selezionare Start execution (Avvia esecuzione).
   4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

      Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Esempio di codice della macchina a stati

La macchina a stati di questo progetto di esempio si integra con SageMaker e AWS Lambda passando i parametri direttamente a tali risorse e utilizza un bucket Amazon S3 per l'origine e l'output dei dati di addestramento.

Sfoglia questo esempio di macchina a stati per vedere come Step Functions controlla Lambda e. SageMaker

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consultaUtilizzo AWS Step Functions con altri servizi.

```
{
  "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
    }
  }
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Esempio IAM

Questi esempi di policy AWS Identity and Access Management (IAM) generati dal progetto di esempio includono il privilegio minimo necessario per eseguire la macchina a stati e le risorse correlate. Ti consigliamo di includere solo le autorizzazioni necessarie nelle tue policy IAM.

```
{
    "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"
        }
    ]
}
```

La seguente policy consente alla funzione Lambda di seminare il bucket Amazon S3 con dati di esempio.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::trainandbatchtransform-s3bucket-1jn1le6gadwfz/*",
            "Effect": "Allow"
        }
    ]
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Ottimizzazione di un modello di Machine Learning.

Questo progetto di esempio dimostra come ottimizzare SageMaker gli iperparametri di un modello di machine learning e trasformare in batch un set di dati di test.

In questo progetto, Step Functions utilizza una funzione Lambda per seminare un bucket Amazon S3 con un set di dati di test. Quindi crea un processo di ottimizzazione degli iperparametri utilizzando l'integrazione del servizio. SageMaker Quindi utilizza una funzione Lambda per estrarre il percorso

dei dati, salva il modello di ottimizzazione, estrae il nome del modello e quindi esegue un processo di trasformazione in batch per eseguire l'inferenza. SageMaker

Per ulteriori informazioni sulle SageMaker integrazioni dei servizi Step Functions, consulta quanto segue:

- Utilizzo AWS Step Functions con altri servizi
- Gestisci SageMaker con Step Functions

> **Note**
>
> Questo progetto di esempio potrebbe comportare costi.
> Per i nuovi AWS utenti, è disponibile un livello di utilizzo gratuito. In questo piano, i servizi sono gratuiti al di sotto di un determinato livello di utilizzo. Per ulteriori informazioni sui AWS costi e sul piano gratuito, consulta la sezione SageMakerPrezzi.

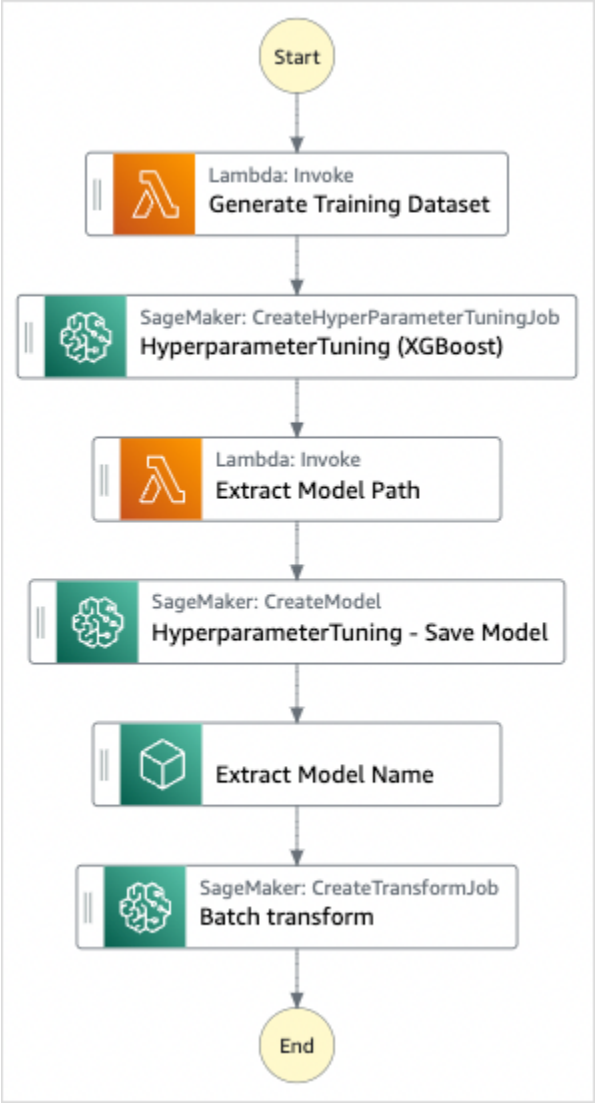
Fase 1: Creazione della macchina a stati e fornitura delle risorse

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`Tune a machine learning model`** nella casella di ricerca, quindi scegli Ottimizza un modello di machine learning dai risultati di ricerca che vengono restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Questo progetto di esempio utilizza le seguenti risorse:

   - Tre funzioni AWS Lambda
   - Un bucket Amazon Simple Storage Service (Amazon S3).
   - Una macchina AWS Step Functions statale
   - Ruoli correlati AWS Identity and Access Management (IAM)

L'immagine seguente mostra il grafico del flusso di lavoro per il progetto di esempio Tune a machine learning model:



5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

     In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione

Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

> **Important**
>
> Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

- Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

  > **Tip**
  >
  > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

  Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

  La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Mentre le risorse vengono distribuite, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

  Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

  > **Important**
  >
  > Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.

2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

   1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

      > **Note**
      >
      > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

   2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

      Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

      > **Note**
      >
      > Se il progetto demo che hai distribuito contiene dati di input di esecuzione precompilati, usa quell'input per eseguire la macchina a stati.

   3. Selezionare Start execution (Avvia esecuzione).
   4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

      Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vedere Pagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Esempio di codice della macchina a stati

La macchina a stati di questo progetto di esempio si integra con SageMaker e AWS Lambda passando i parametri direttamente a tali risorse e utilizza un bucket Amazon S3 per l'origine e l'output dei dati di addestramento.

Sfoglia questo esempio di macchina a stati per vedere come Step Functions controlla Lambda e. SageMaker

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consultaUtilizzo AWS Step Functions con altri servizi.

```
{
    "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
        }
    }
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Esempi IAM

Questi esempi di policy AWS Identity and Access Management (IAM) generati dal progetto di esempio includono il minimo privilegio necessario per eseguire la macchina a stati e le risorse correlate. Ti consigliamo di includere solo le autorizzazioni necessarie nelle tue policy IAM.

La seguente policy IAM è collegata alla macchina a stati e consente all'esecuzione della macchina a stati di accedere alle risorse necessarie SageMaker, Lambda e 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"
        }
    ]
}
```

La seguente politica IAM è referenziata nei `HyperparameterTuning` campi `TrainingJobDefinition` e dello stato. `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"
        }
    ]
}
```

La seguente policy IAM consente alla funzione Lambda di seminare il bucket Amazon S3 con dati di esempio.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::stepfunctionssample-sagemak-
bucketformodelanddata-80fblmdlcs9f/*",
            "Effect": "Allow"
        }
    ]
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Elabora messaggi ad alto volume da Amazon SQS (Express Workflows)

Questo progetto di esempio dimostra come utilizzare un AWS Step Functions Express Workflow per elaborare messaggi o dati da una fonte di eventi ad alto volume, come Amazon Simple Queue Service (Amazon SQS). Dal momento che i flussi di lavoro Express possono essere avviati a una velocità molto elevata, sono ideali per i carichi di lavoro di elaborazione eventi o di streaming dei dati a volume elevato.

Di seguito sono indicati due metodi utilizzati comunemente per eseguire la macchina a stati da un'origine evento:

- Configura una regola Amazon CloudWatch Events per avviare l'esecuzione di una macchina a stati ogni volta che l'origine dell'evento emette un evento. Per ulteriori informazioni, consulta Creazione di una regola per CloudWatch gli eventi che si attiva su un evento.
- Mappare l'origine evento a una funzione Lambda e scrivere codice di funzione per eseguire la macchina a stati. La AWS Lambda funzione viene richiamata ogni volta che l'origine dell'evento emette un evento, avviando a sua volta l'esecuzione di una macchina a stati. Per ulteriori informazioni, consulta Using AWS Lambda with Amazon SQS.

Questo progetto di esempio utilizza il secondo metodo per avviare un'esecuzione ogni volta che la coda Amazon SQS invia un messaggio. Puoi utilizzare una configurazione simile per attivare l'esecuzione di Express Workflows da altre fonti di eventi, come Amazon Simple Storage Service (Amazon S3), Amazon DynamoDB e Amazon Kinesis.

Per ulteriori informazioni sulle integrazioni dei servizi Express Workflows e Step Functions, consulta quanto segue:

- Flussi di lavoro Standard e Express
- Utilizzo AWS Step Functions con altri servizi
- Quote

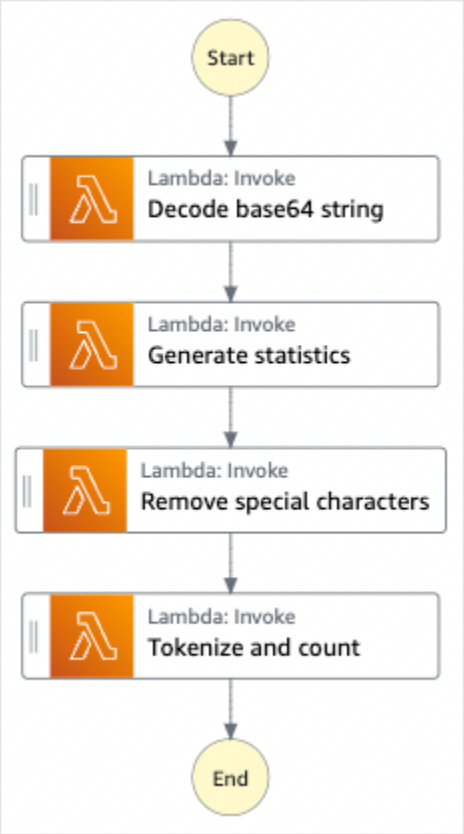
Fase 1: Creare la macchina a stati e fornire risorse

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digitate **`Process high-volume messages from SQS`** nella casella di ricerca, quindi scegliete Elabora messaggi ad alto volume da SQS tra i risultati di ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Questo progetto di esempio utilizza le seguenti risorse:

   - Quattro funzioni Lambda
   - Una coda Amazon SQS
   - Una macchina a AWS Step Functions stati
   - Ruoli correlati AWS Identity and Access Management (IAM)

   L'immagine seguente mostra il grafico del flusso di lavoro per il progetto di esempio Process High Volume Messages from SQS:



5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

     In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

     > **Important**
     >
     > Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

- Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

  > **Tip**
  >
  > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

  Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

  La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Mentre le risorse vengono distribuite, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

  Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

  > **Important**
  >
  > Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Attivare l'esecuzione della macchina a stati

1. Apri la console Amazon SQS.
2. Selezionare la coda creata dal progetto di esempio.

   Il nome sarà simile a Example-SQSQueue-wJalrXUtnFEMI.
3. Dall'elenco Queue Actions (Azioni coda), selezionare Send a Message (Invia un messaggio).
4. Utilizzare il pulsante di copia per copiare il seguente messaggio, quindi immetterlo nella finestra Send a Message (Invia un messaggio) e scegliere Send Message (Invia messaggio).

> **Note**
>
> In questo messaggio di esempio sono state inserite interruzioni di riga nella riga `input:` per adattarla alla pagina. Utilizzare il pulsante di copia o assicurarsi in altro modo che venga immesso come singola riga senza interruzioni.

```
{
      "input":
 "QW5kIGxpa2UgdGhlIGJhc2VsZXNzIGZhYnJpYyBvZiB0aGlzIHZpc2lvbiwgVGhlIGNsb3VkLWNhcHBlZCB0b3dlc

  91cyBwYWxhY2VzLCBUaGUgc29sZW1uIHRlbXBsZXMsIHRoZSBncmVhdCBnbG9iZSBpdHNlbGbigJQgWWVhLCBhbGw

  ZXJpdOKAlHNoYWxsIGRpc3NvbHZlLCBBbmQgbGlrZSB0aGlzIGluc3Vic3RhbnRpYWwgcGFnZWFudCBmYWRlZCwgT

  FjayBiZWhpbmQuIFdlIGFyZSBzdWNoIHN0dWZmIEFzIGRyZWFtcyBhcmUgbWFkZSBvbiwgYW5kIG91ciBsaXR0bGU

  ZGVkIHdpdGggYSBzbGVlcC4gU2lyLCBJIGFtIHZleGVkLiBCZWFyIHdpdGggbXkgd2Vha25lc3MuIE15IG9sZCBic

  xlZC4gQmUgbm90IGRpc3R1cmJlZCB3aXRoIG15IGluZmlybWl0eS4gSWYgeW91IGJlIHBsZWFzZWQsIHJldGlyZSB

  QW5kIHRoZXJlIHJlcG9zZS4gQSB0dXJuIG9yIHR3byBJ4oCZbGwgd2FsayBUbyBzdGlsbCBteSBzdWF0aW5nIG1pb
}
```

5. Scegli Chiudi.
6. Apri la console Step Functions.
7. Vai al tuo gruppo di CloudWatch log Amazon Logs e ispeziona i log. Il nome del gruppo di log sarà simile all'esempio- ExpressLogGroup -WJalRxUtnFemi.

Esempio di codice di funzione Lambda

Di seguito è riportato il codice della funzione Lambda che mostra come la funzione Lambda di avvio avvia l'esecuzione di una macchina a stati utilizzando l'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
    )
```

Esempio di codice della macchina a stati

Il flusso di lavoro Express in questo progetto di esempio consiste in un insieme di funzioni Lambda per l'elaborazione del testo.

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consulta. Utilizzo AWS Step Functions con altri servizi

```
{
  "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
    }
  }
}
```

Esempio IAM

Questo esempio di policy AWS Identity and Access Management (IAM) generato dal progetto di esempio include il minimo privilegio necessario per eseguire la macchina a stati e le risorse correlate. Ti consigliamo di includere solo le autorizzazioni necessarie nelle tue policy IAM.

```
{
    "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"
        }
```

```
    ]
}
```

La seguente politica garantisce che vi siano autorizzazioni sufficienti per i log. CloudWatch

```
{
    "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"
        }
    ]
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Esempio di checkpoint selettivo (flussi di lavoro Express)

Questo progetto di esempio illustra come combinare i flussi di lavoro Standard ed Express eseguendo un flusso di lavoro di e-commerce simulato che effettua il checkpoint selettivo. La distribuzione di questo progetto di esempio crea una macchina a stati Standard Workflows, una macchina a stati Express Workflows annidata, una AWS Lambda funzione, una coda Amazon Simple Queue Service (Amazon SQS) e un argomento Amazon Simple Notification Service (Amazon SNS).

Per ulteriori informazioni su Express Workflows, flussi di lavoro annidati e integrazioni dei servizi Step Functions, consulta quanto segue:

- Flussi di lavoro Standard e Express
- Avvio delle esecuzioni di flusso di lavoro da uno stato di task
- Utilizzo AWS Step Functions con altri servizi

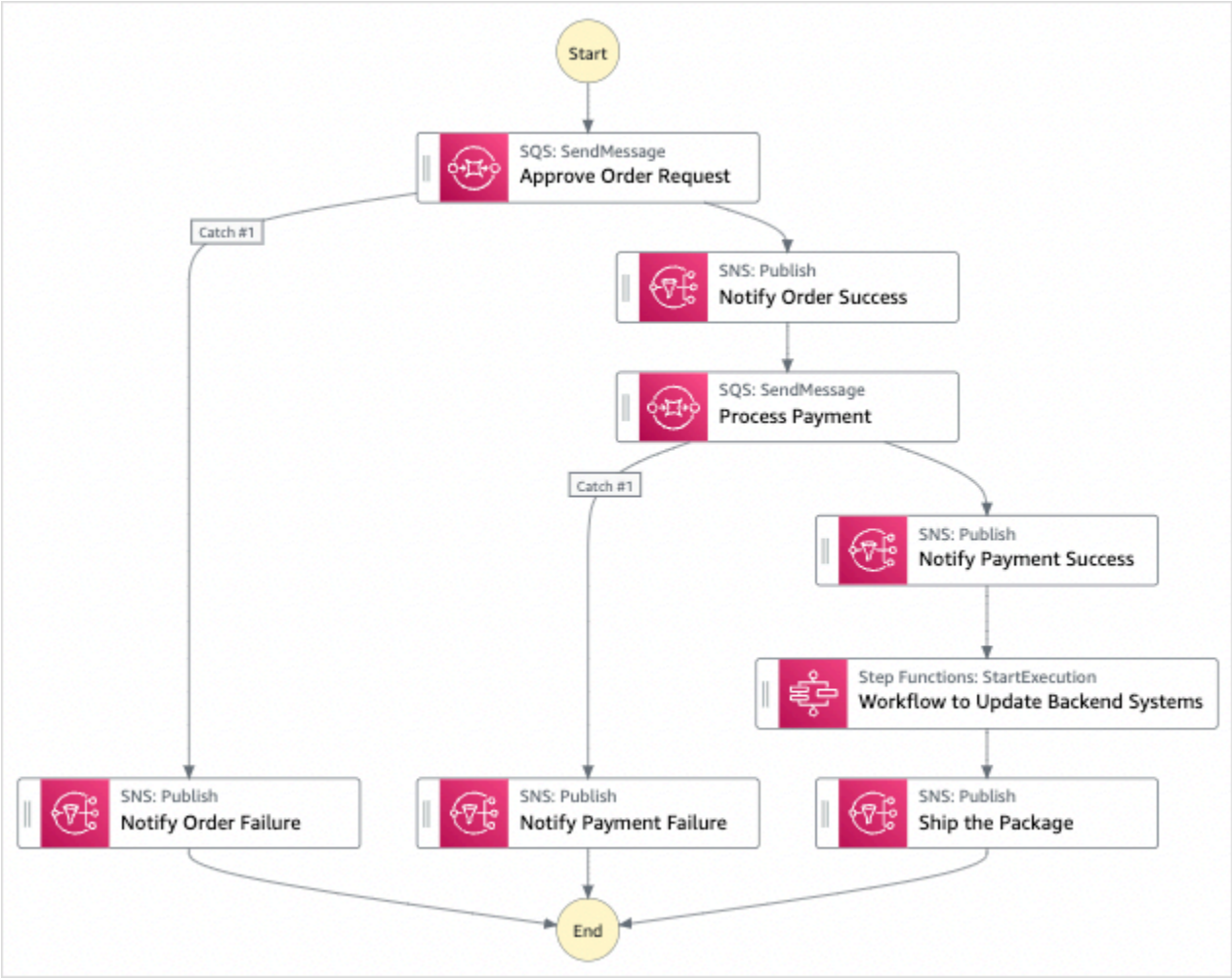
Fase 1: Creare la State Machine e fornire risorse

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`Selective checkpointing example`** nella casella di ricerca, quindi scegli Esempio di checkpoint selettivo dai risultati della ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Questo progetto di esempio utilizza le seguenti risorse:

   - Una funzione AWS Lambda
   - Una coda Amazon SQS
   - Un argomento Amazon SNS
   - Una macchina a AWS Step Functions stati di tipo Standard
   - Una macchina a stati Step Functions di tipo Express
   - Ruoli correlati AWS Identity and Access Management (IAM)

   L'immagine seguente mostra il grafico del flusso di lavoro per il progetto di esempio di checkpointing selettivo:



5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

     In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

> **Important**
>
> Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

- Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

  > **Tip**
  >
  > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

  Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

  La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Mentre le risorse vengono distribuite, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

  Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

  > **Important**
  >
  > Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Dopo aver distribuito le risorse del progetto di esempio, eseguire le operazioni seguenti.

Fase 2: Eseguire la macchina a stati

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.
2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.

3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

   1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

      > **Note**
      >
      > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

   2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

      Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

      > **Note**
      >
      > Se il progetto demo che hai distribuito contiene dati di input di esecuzione precompilati, usa quell'input per eseguire la macchina a stati.

   3. Selezionare Start execution (Avvia esecuzione).
   4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

      Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

4. Vai al gruppo CloudWatch Logs log e ispeziona i log. Il nome del gruppo di log sarà simile all'esempio- ExpressLogGroup -WJalRxUtnFemi.

Codice macchina a stati di esempio per il padre (flussi di lavoro standard)

La macchina a stati di questo progetto di esempio si integra con i flussi di lavoro Amazon SQS, Amazon SNS e Step Functions Express.

Sfoglia questo esempio di macchina a stati per vedere come Step Functions elabora l'input da Amazon SQS e Amazon SNS, quindi utilizza una macchina a stati Express Workflows annidata per aggiornare i sistemi di backend.

Per ulteriori informazioni su come controllare altri servizi, AWS Step Functions consulta. AWS Utilizzo AWS Step Functions con altri servizi

```
{
    "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
        }
    }
}
```

Esempio di ruolo IAM per la Parent State Machine

Questi esempi di policy AWS Identity and Access Management (IAM) generati dal progetto di esempio includono il privilegio minimo necessario per eseguire la macchina a stati e le risorse correlate. Ti consigliamo di includere solo le autorizzazioni necessarie nelle tue policy IAM.

Politica di Amazon SNS:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": "arn:aws:sns:us-east-1:123456789012:Checkpoint-SNSTopic-
wJalrXUtnFEMI",
            "Effect": "Allow"
        }
    ]
}
```

Politica di Amazon SQS:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sqs:SendMessage"
            ],
            "Resource": "arn:aws:sqs:us-east-1:123456789012:Checkpoint-SQSQueue-
je7MtGbClwBF",
            "Effect": "Allow"
        }
    ]
}
```

Policy di esecuzione degli stati:

```
{
    "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"
        }
    ]
}
```

Codice macchina a stati di esempio per la macchina a stati nidificato (flussi di lavoro Express)

La macchina a stati in questo progetto di esempio aggiorna le informazioni di back-end quando viene chiamata dalla macchina a stati padre.

Sfoglia questo esempio di macchina a stati per vedere come Step Functions aggiorna i diversi componenti dei finti sistemi di backend per l'e-commerce.

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consulta. Utilizzo AWS Step Functions con altri servizi

```
{
  "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
    }
  }
}
```

Esempio di ruolo IAM per Child State Machine

Questo esempio di policy AWS Identity and Access Management (IAM) generato dal progetto di esempio include il minimo privilegio necessario per eseguire la macchina a stati e le risorse correlate. Ti consigliamo di includere solo le autorizzazioni necessarie nelle tue policy IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:123456789012:function:Example-
UpdateDatabaseLambdaFunction-wJalrXUtnFEMI"
```

```
            ],
            "Effect": "Allow"
        }
    ]
}
```

La seguente politica garantisce che vi siano autorizzazioni sufficienti per i 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"
        }
    ]
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Crea un AWS CodeBuild progetto (CodeBuild, Amazon SNS)

Questo progetto di esempio dimostra come AWS Step Functions creare un AWS CodeBuild progetto, eseguire test e quindi inviare una notifica Amazon SNS.

Fase 1: Creare la macchina a stati e fornire risorse

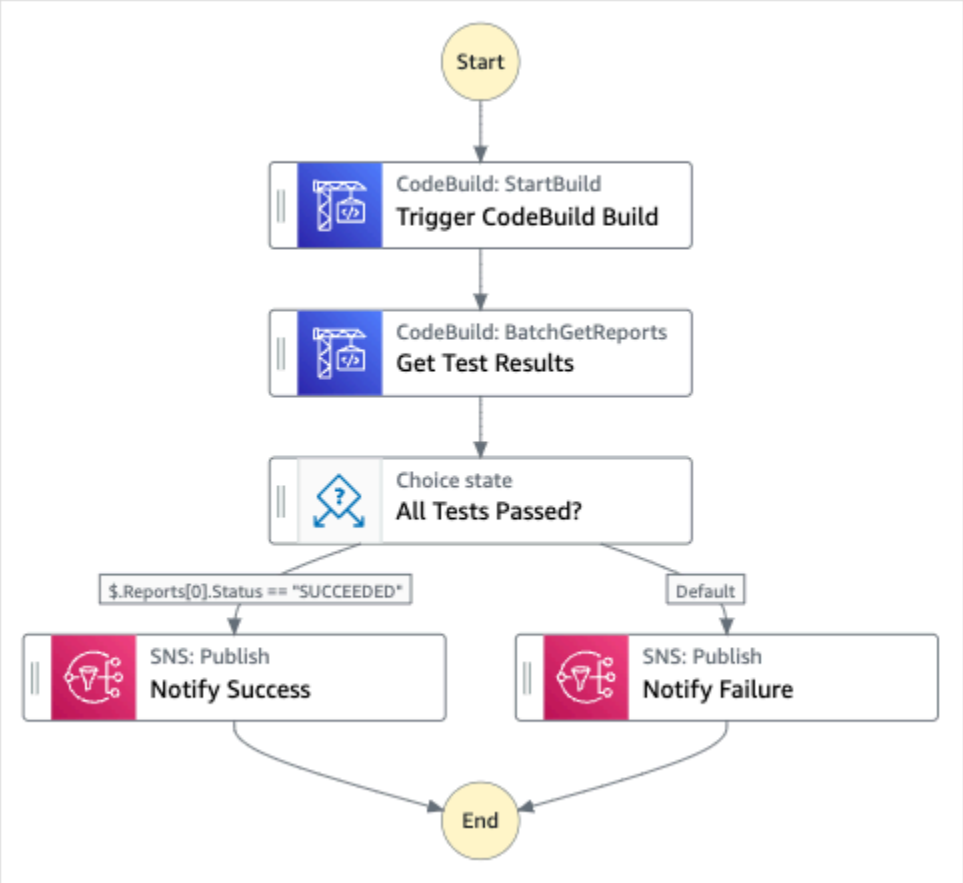
1. Apri la console Step Functions e scegli Crea macchina a stati.

2. Digita **`Start a CodeBuild build`** nella casella di ricerca, quindi scegli Avvia una CodeBuild build dai risultati della ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Questo progetto di esempio utilizza le seguenti risorse:

   - Una AWS CodeBuild build
   - Un argomento Amazon SNS
   - Una macchina AWS Step Functions statale
   - Ruoli correlati AWS Identity and Access Management (IAM)

   L'immagine seguente mostra il grafico del flusso di lavoro per il progetto di esempio Start a CodeBuild build:



5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo di workflow per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

     In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

     > **Important**
     >
     > Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

   - Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

     > **Tip**
     >
     > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

     Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

     La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Durante la distribuzione delle risorse, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

     Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

> **Important**
>
> Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.
2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

    1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

        > **Note**
        >
        > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

    2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

        Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

        > **Note**
        >
        > Se il progetto demo che hai distribuito contiene dati di input di esecuzione precompilati, usa quell'input per eseguire la macchina a stati.

    3. Selezionare Start execution (Avvia esecuzione).
    4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile

esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Esempio di codice della macchina a stati

La macchina a stati di questo progetto di esempio si integra con Amazon CodeBuild SNS.

Sfoglia questo esempio di macchina a stati per vedere come Step Functions utilizza una macchina a stati per creare un CodeBuild progetto, quindi invia un argomento di Amazon SNS con un messaggio sull'esito positivo o negativo del processo.

Per ulteriori informazioni su come Step Functions può controllare altri AWS servizi, vedereUtilizzo AWS Step Functions con altri servizi.

```
{
  "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
    }
  }
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Preelabora i dati e addestra un modello di machine learning

Questo progetto di esempio dimostra come utilizzare SageMaker e preelaborare i dati e AWS Step Functions addestrare un modello di apprendimento automatico.

In questo progetto, Step Functions utilizza una funzione Lambda per seminare un bucket Amazon S3 con un set di dati di test e uno script Python per l'elaborazione dei dati. Quindi addestra un modello di machine learning ed esegue una trasformazione in batch, utilizzando l'integrazione del servizio. SageMaker

Per ulteriori informazioni sulle SageMaker integrazioni dei servizi Step Functions, consulta quanto segue:

- Utilizzo AWS Step Functions con altri servizi
- Gestisci SageMaker con Step Functions

> **Note**
>
> Questo progetto di esempio potrebbe comportare costi.
> Per i nuovi AWS utenti, è disponibile un livello di utilizzo gratuito. In questo piano, i servizi sono gratuiti al di sotto di un determinato livello di utilizzo. Per ulteriori informazioni sui AWS costi e sul piano gratuito, consulta la sezione SageMaker Prezzi.

Fase 1: Creazione della macchina a stati e fornitura delle risorse

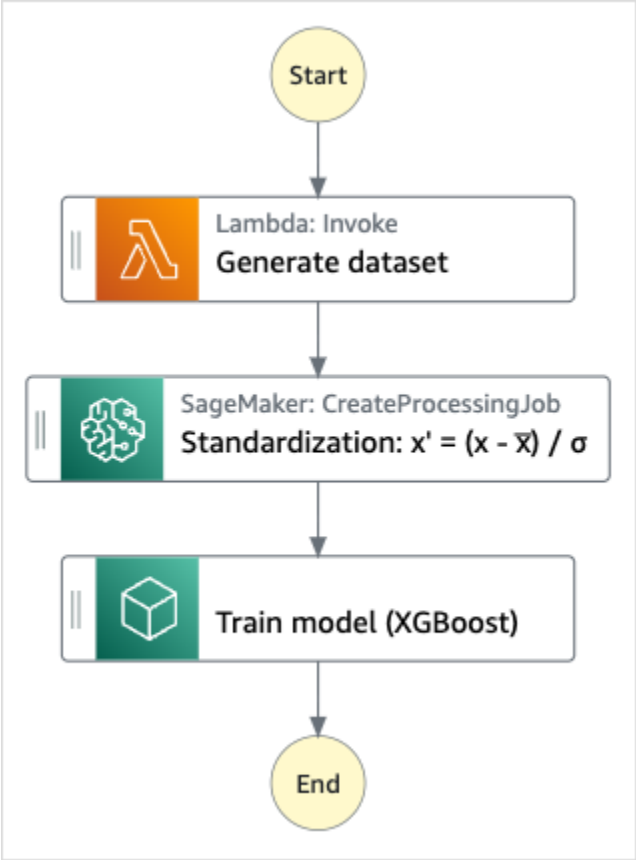
1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`Preprocess data and train a machine learning model`** nella casella di ricerca, quindi scegli Preelabora i dati e addestra un modello di apprendimento automatico dai risultati della ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Questo progetto di esempio utilizza le seguenti risorse:

   - Una funzione AWS Lambda
   - Un bucket Amazon S3
   - Una macchina AWS Step Functions statale

- Ruoli correlati AWS Identity and Access Management (IAM)

L'immagine seguente mostra il grafico del flusso di lavoro per i dati di preelaborazione e addestra un progetto di esempio di modello di apprendimento automatico:



5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

     In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

> **Important**
>
> Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

- Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

  > **Tip**
  >
  > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

  Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

  La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Mentre le risorse vengono distribuite, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

  Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

  > **Important**
  >
  > Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.
2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

   > **Note**
   >
   > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

   Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

   > **Note**
   >
   > Se il progetto demo che hai distribuito contiene dati di input di esecuzione precompilati, usa quell'input per eseguire la macchina a stati.

3. Selezionare Start execution (Avvia esecuzione).
4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

   Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Esempio di codice della macchina a stati

La macchina a stati di questo progetto di esempio si integra con SageMaker e AWS Lambda passando i parametri direttamente a tali risorse e utilizza un bucket Amazon S3 per l'origine e l'output dei dati di addestramento.

Sfoglia questo esempio di macchina a stati per vedere come Step Functions controlla Lambda e. SageMaker

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consultaUtilizzo AWS Step Functions con altri servizi.

```
{
  "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
    }
  }
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Esempio IAM

Questi esempi di policy AWS Identity and Access Management (IAM) generati dal progetto di esempio includono il privilegio minimo necessario per eseguire la macchina a stati e le risorse correlate. Ti consigliamo di includere solo le autorizzazioni necessarie nelle tue policy IAM.

```
{
    "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"
        }
    ]
}
```

La seguente policy consente alla funzione Lambda di seminare il bucket Amazon S3 con dati di esempio.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::featuretransform-
bucketforcodeanddata-1jn1le6gadwfz/*",
            "Effect": "Allow"
        }
    ]
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Esempio di orchestrazione Lambda

Questo progetto di esempio dimostra come integrare AWS Lambda le funzioni nelle macchine a stati Step Functions.

In questo progetto, Step Functions utilizza le funzioni Lambda per controllare il prezzo di un'azione e determinare una raccomandazione commerciale di acquisto o vendita. L'utente riceve quindi questa raccomandazione e può scegliere se acquistare o vendere il titolo. Il risultato dello scambio viene restituito utilizzando un argomento SNS.

Per ulteriori informazioni sulle integrazioni dei servizi Step Functions, consulta quanto segue:

- Utilizzo AWS Step Functions con altri servizi
- Politiche IAM per:
  - Politiche IAM per AWS Lambda
  - Politiche IAM per Amazon SQS
  - Politiche IAM per Amazon SNS

> **Note**
>
> Questo progetto di esempio potrebbe comportare costi.

Per i nuovi AWS utenti, è disponibile un livello di utilizzo gratuito. In questo piano, i servizi sono gratuiti al di sotto di un determinato livello di utilizzo. Per ulteriori informazioni sui AWS costi e sul piano gratuito, consulta la sezione Prezzi.

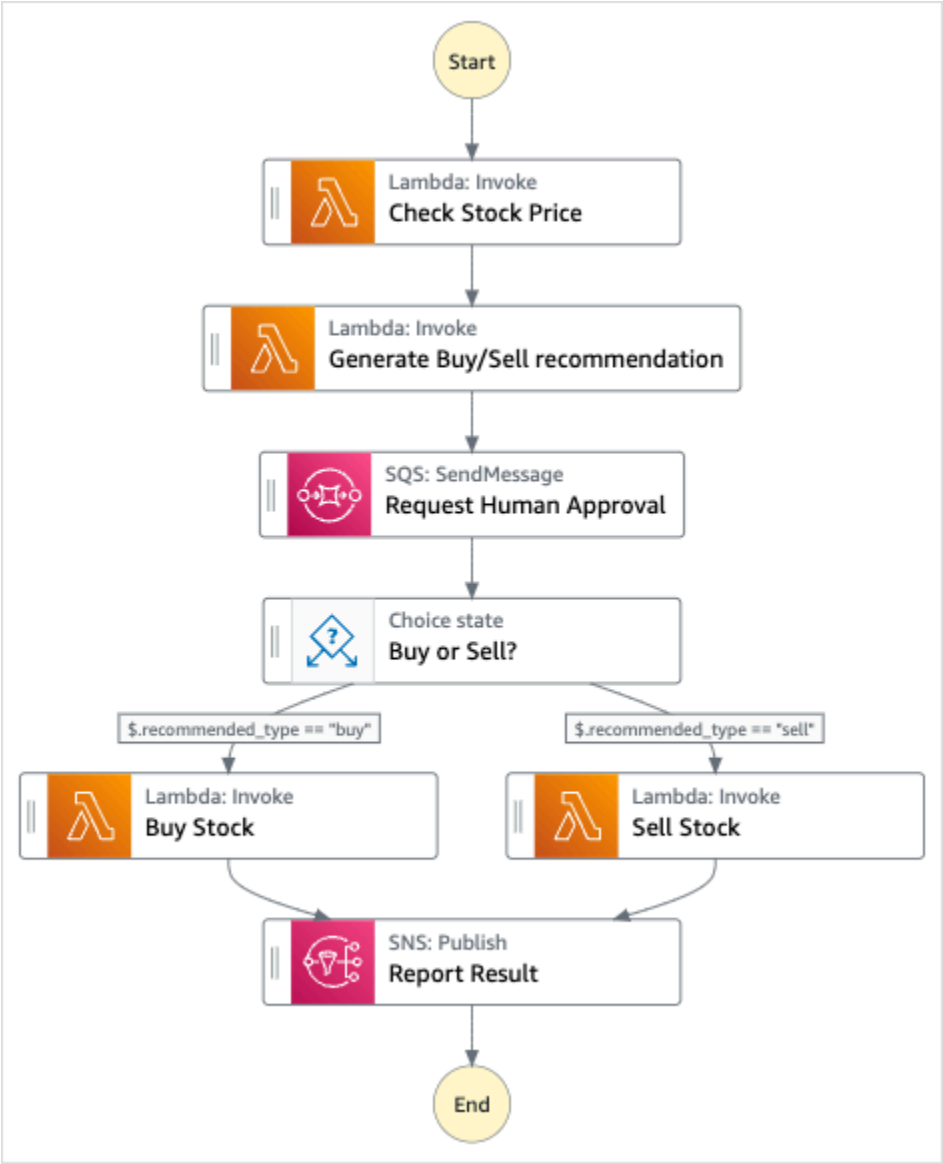
Fase 1: Creare la macchina a stati e fornire risorse

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`Orchestrate Lambda functions`** nella casella di ricerca, quindi scegli Orchestrate Lambda functions dai risultati di ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Questo progetto di esempio utilizza le seguenti risorse:

   - Cinque funzioni Lambda
   - Una coda Amazon Simple Queue Service
   - Un argomento Amazon Simple Notification Service
   - Una macchina a stati AWS Step Functions
   - Ruoli AWS Identity and Access Management (IAM) correlati

   L'immagine seguente mostra il grafico del flusso di lavoro per il progetto di esempio delle Lambdafunzioni di Orchestrate:



5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

     In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step

Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

> **Important**
>
> Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

- Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

  > **Tip**
  >
  > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

  Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

  La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Durante la distribuzione delle risorse, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

  Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

  > **Important**
  >
  > Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

Dopo aver fornito e distribuito tutte le risorse, viene visualizzata la finestra di dialogo Avvia esecuzione.

1. Nella pagina State machines, scegliete il vostro progetto di esempio.
2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

    1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

        > **Note**
        >
        > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

    2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

        Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

        > **Note**
        >
        > Se il progetto demo che hai distribuito contiene dati di input di esecuzione precompilati, usa quell'input per eseguire la macchina a stati.

    3. Selezionare Start execution (Avvia esecuzione).
    4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

        Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Informazioni sulla macchina a stati e sulla sua esecuzione

La macchina a stati di questo progetto di esempio si integra AWS Lambda passando i parametri direttamente a tali risorse, utilizza una coda Amazon SQS per gestire la richiesta di approvazione umana e utilizza un argomento Amazon SNS per restituire i risultati della query.

Un'esecuzione di Step Functions riceve un testo JSON come input e lo passa al primo stato del flusso di lavoro. I singoli stati ricevono i dati JSON come input e in genere passano i dati JSON come output allo stato successivo. In questo progetto di esempio, l'output di ogni passaggio viene passato come input alla fase successiva del flusso di lavoro. Ad esempio, la fase di raccomandazione Generate Buy/Sell riceve come input l'output della fase Check Stock Price. Inoltre, l'output della fase di raccomandazione Generate Buy/Sell viene passato come input alla fase successiva, Request Human Approval, che simula una fase di approvazione umana.

> **Note**
>
> Per visualizzare l'output restituito da un passaggio e l'input passato a un passaggio, apri la pagina Dettagli di esecuzione per l'esecuzione del flusso di lavoro. Nella sezione Dettagli del passaggio, visualizza l'input e l'output per ogni passaggio selezionato nella modalità Visualizza.

Per implementare una fase di approvazione umana, in genere si sospende l'esecuzione del flusso di lavoro fino alla restituzione di un token di attività. In questo progetto di esempio, un messaggio viene passato a una coda Amazon SQS, che viene utilizzata come trigger per la funzione Lambda definita per gestire la funzionalità di callback. Il messaggio contiene un token di attività e l'output restituito dal passaggio precedente. La funzione Lambda viene richiamata con il payload del messaggio. L'esecuzione del flusso di lavoro viene sospesa fino a quando non riceve il token dell'attività con una chiamata API. `SendTaskSuccess` Per ulteriori informazioni sui task token, consulta. Attendere un callback con il token dell'attività

Il codice seguente per la `StepFunctionsSample-HelloLambda-ApproveSqsLambda` funzione mostra come viene definita l'approvazione automatica di qualsiasi attività inviata dalla coda di Amazon SQS tramite la macchina a stati Step Functions.

Esempio di codice della funzione Lambda per gestire la funzionalità di callback e restituire il token dell'attività

```
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);
      });
  }
};
```

Sfoglia questo esempio di macchina a stati per vedere come Step Functions controlla Lambda e Amazon SQS.

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consulta. Utilizzo AWS Step Functions con altri servizi

```
{
    "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
        }
    }
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Esempi IAM

Questi esempi di policy AWS Identity and Access Management (IAM) generati dal progetto di esempio includono il minimo privilegio necessario per eseguire la macchina a stati e le risorse correlate. Ti consigliamo di includere solo le autorizzazioni necessarie nelle tue policy IAM.

```
{
    "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"
        }
    ]
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Avvia una query su Athena

Questo progetto di esempio, basato su flussi di lavoro standard, dimostra come utilizzare Step Functions e Amazon Athena per avviare una query Athena e inviare una notifica con i risultati delle query.

In questo progetto, Step Functions utilizza le funzioni Lambda e un AWS Glue crawler per generare una serie di dati di esempio. Quindi esegue una query utilizzando l'integrazione del servizio Athena e restituisce i risultati utilizzando un argomento SNS.

Per ulteriori informazioni sulle integrazioni dei servizi Athena e Step Functions, consulta quanto segue:

- Utilizzo AWS Step Functions con altri servizi
- Chiama Athena con Step Functions

> **Note**
>
> Questo progetto di esempio potrebbe comportare costi.
> Per i nuovi AWS utenti, è disponibile un livello di utilizzo gratuito. In questo piano, i servizi sono gratuiti al di sotto di un determinato livello di utilizzo. Per ulteriori informazioni sui AWS costi e sul piano gratuito, consulta i prezzi di Athena.

Fase 1: Creazione della macchina a stati e fornitura delle risorse

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`Start an Athena query`** nella casella di ricerca, quindi scegli Avvia una Athena query dai risultati della ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Questo progetto di esempio utilizza le seguenti risorse:

   - Un'Amazon Athenainterrogazione
   - Un Crawler di AWS Glue
   - Un argomento Amazon SNS
   - Una macchina a stati AWS Step Functions
   - Ruoli AWS Identity and Access Management (IAM) correlati

   L'immagine seguente mostra il grafico del flusso di lavoro per il progetto di esempio Start an Athena query:

5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

     In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

> **Important**
>
> Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

- Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

  > **Tip**
  >
  > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

  Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

  La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Durante la distribuzione delle risorse, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

  Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

  > **Important**
  >
  > Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.
2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

   > **Note**
   >
   > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

   Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

   > **Note**
   >
   > Se il progetto demo che hai distribuito contiene dati di input di esecuzione precompilati, usa quell'input per eseguire la macchina a stati.

3. Selezionare Start execution (Avvia esecuzione).
4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

   Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Esempio di codice della macchina a stati

La macchina a stati di questo progetto di esempio si integra con Athena AWS Lambda e passa i parametri direttamente a tali risorse e utilizza un argomento SNS per restituire i risultati della query.

Sfoglia questo esempio di macchina a stati per vedere come Step Functions controlla Lambda e Athena.

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consulta. Utilizzo AWS Step Functions con altri servizi

```
{
  "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
    }
  }
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Esempio IAM

Questi esempi di policy AWS Identity and Access Management (IAM) generati dal progetto di esempio includono il privilegio minimo necessario per eseguire la macchina a stati e le risorse correlate. Ti consigliamo di includere solo le autorizzazioni necessarie nelle tue policy IAM.

```
{
    "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"
        }
    ]
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Esegui più query (Amazon Athena, Amazon SNS)

Questo progetto di esempio dimostra come eseguire le query Athena in successione e poi in parallelo, gestire gli errori e quindi inviare una notifica Amazon SNS a seconda che le query abbiano esito positivo o negativo.

In questo progetto, Step Functions utilizza una macchina a stati per eseguire le query Athena in modo sincrono. Dopo la restituzione dei risultati della query, immettete lo stato parallelo con due query Athena eseguite in parallelo. Attende quindi che il processo abbia esito positivo o negativo e invia un argomento di Amazon SNS con un messaggio indicante se il processo è riuscito o meno.

Fase 1: Creare la macchina a stati e fornire risorse

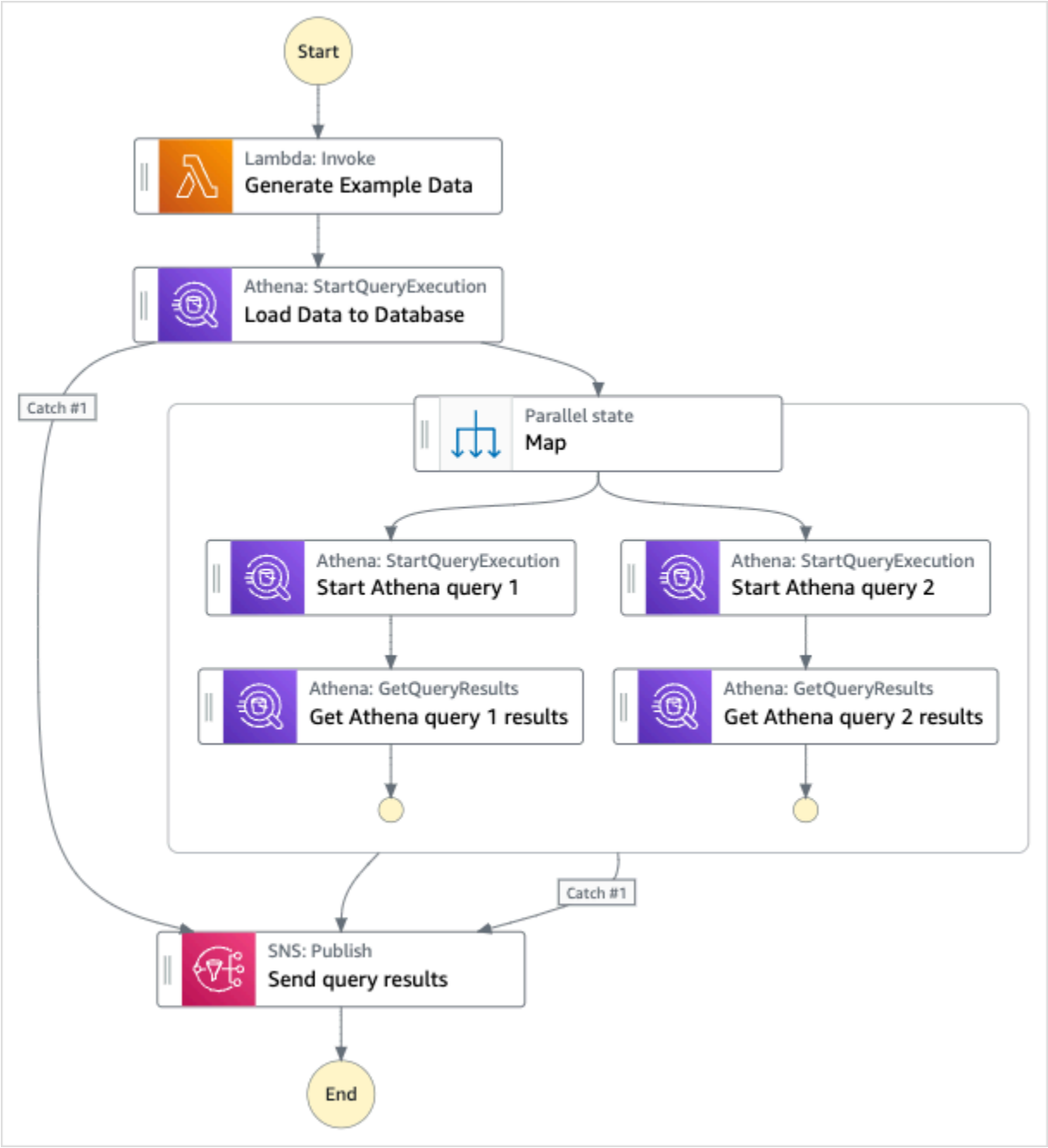
1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`Execute multiple queries`** nella casella di ricerca, quindi scegli Esegui più interrogazioni dai risultati della ricerca restituiti.
3. Seleziona Successivo per continuare.

4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Questo progetto di esempio utilizza le seguenti risorse:

   - Amazon Athenainterrogazioni
   - Un argomento Amazon SNS
   - Una macchina a stati AWS Step Functions
   - Ruoli AWS Identity and Access Management (IAM) correlati

   L'immagine seguente mostra il grafico del flusso di lavoro per il progetto di esempio Execute multiple queries:



5. Scegliete Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

     In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello

che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

> **Important**
>
> Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

- Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

  > **Tip**
  >
  > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

  Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

  La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Mentre le risorse vengono distribuite, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

  Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

  > **Important**
  >
  > Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.
2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

    1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

        > **Note**
        >
        > Step Functions consente di creare nomi per macchine a stati, esecuzioni, attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

    2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

        Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

    3. Selezionare Start execution (Avvia esecuzione).
    4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

        Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Esempio di codice della macchina a stati

La macchina a stati di questo progetto di esempio si integra con Amazon Athena e Amazon SNS passando i parametri direttamente a tali risorse.

Sfoglia questo esempio di macchina a stati per vedere come Step Functions controlla Amazon Athena e Amazon SNS connettendosi all'Amazon Resource Name (ARN) sul campo e passando `Resource` `Parameters` all'API del servizio.

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consulta. Utilizzo AWS Step Functions con altri servizi

```
{
    "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
      }
    }
  }
```

Esempi IAM

Questo esempio di policy AWS Identity and Access Management (IAM) generato dal progetto di esempio include il minimo privilegio necessario per eseguire la macchina a stati e le risorse correlate. Ti consigliamo di includere solo le autorizzazioni necessarie nelle tue policy IAM.

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

Pubblicazione SNS

```
{
    "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"
            ]
        }
    ]
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Interroga set di dati di grandi dimensioni (Amazon Athena, Amazon S3 e Amazon SNS AWS Glue)

Questo progetto di esempio dimostra come importare un set di dati di grandi dimensioni in Amazon S3 e partizionarlo AWS Glue tramite crawler, quindi eseguire query Amazon Athena su quella partizione.

In questo progetto, la macchina a stati Step Functions richiama un AWS Glue crawler che partiziona un set di dati di grandi dimensioni in Amazon S3. Una volta che il AWS Glue crawler restituisce un messaggio di successo, il flusso di lavoro esegue le query Athena su quella partizione. Una volta completata correttamente l'esecuzione della query, viene inviata una notifica Amazon SNS a un argomento Amazon SNS.

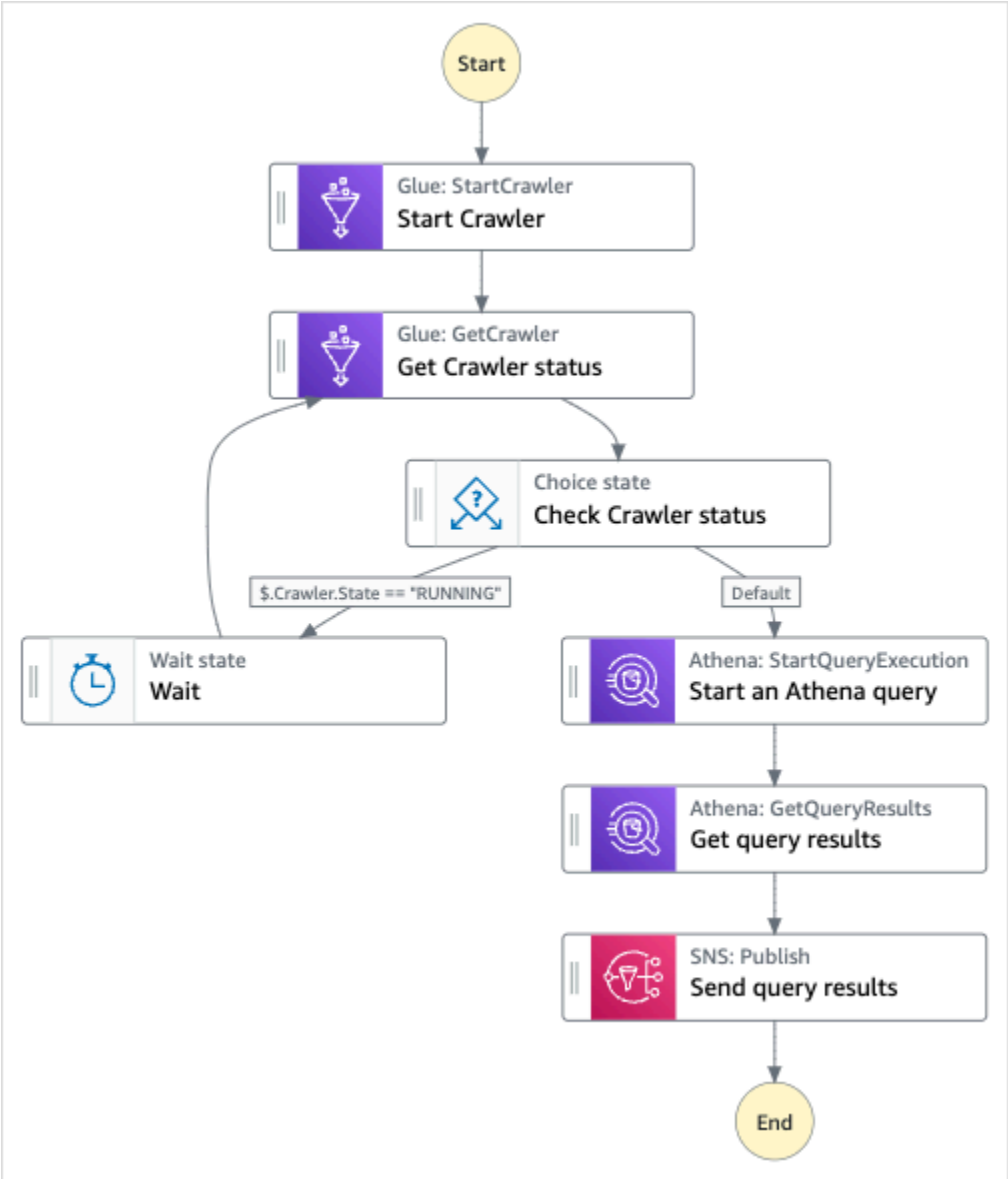
Fase 1: Creare la macchina a stati e fornire risorse

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`Query large datasets`** nella casella di ricerca, quindi scegli Interroga set di dati di grandi dimensioni dai risultati della ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Questo progetto di esempio utilizza le seguenti risorse:

- Un bucket Amazon S3
- Un Crawler di AWS Glue
- Un argomento Amazon SNS
- Una macchina a stati AWS Step Functions
- Ruoli AWS Identity and Access Management (IAM) correlati

L'immagine seguente mostra il grafico del flusso di lavoro per il progetto di esempio Query large datasets:



5. Scegli Usa modello per continuare con la selezione.

6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

     In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

     > **Important**
     >
     > Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

   - Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

     > **Tip**
     >
     > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

     Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

     La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Durante la distribuzione delle risorse, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

     Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

> **Important**
>
> Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.
2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

    1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

        > **Note**
        >
        > Step Functions consente di creare nomi per macchine a stati, esecuzioni, attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

    2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

        Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

    3. Selezionare Start execution (Avvia esecuzione).
    4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

        Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni

sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vedere Pagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Esempio di codice della macchina a stati

La macchina a stati di questo progetto di esempio si integra con Amazon S3 AWS Glue, Amazon Athena e Amazon SNS passando i parametri direttamente a tali risorse.

Sfoglia questo esempio di macchina a stati per vedere come Step Functions controlla Amazon S3 AWS Glue, Amazon Athena e Amazon SNS connettendosi all'Amazon Resource Name (ARN) sul campo e passando `Resource` all'API del servizio. `Parameters`

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consulta. Utilizzo AWS Step Functions con altri servizi

```
{
    "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
      }
    }
  }
```

Esempi IAM

Questi esempi di policy AWS Identity and Access Management (IAM) generati dal progetto di esempio includono il minimo privilegio necessario per eseguire la macchina a stati e le risorse correlate. Ti consigliamo di includere solo le autorizzazioni necessarie nelle tue policy IAM.

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

Pubblicazione SNS

```
{
    "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"
            ]
        }
    ]
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Mantieni aggiornati i dati (Amazon Athena, Amazon S3,) AWS Glue

Questo progetto di esempio dimostra come interrogare una tabella di destinazione per ottenere dati correnti con AWS Glue Catalog, quindi aggiornarla con nuovi dati da altre fonti utilizzando Amazon Athena.

In questo progetto, la macchina a stati Step Functions chiama AWS Glue Catalog per verificare se esiste una tabella di destinazione in un bucket Amazon S3. Se non viene trovata alcuna tabella, verrà creata una nuova tabella. Quindi, Step Functions esegue una query Athena per aggiungere righe alla tabella di destinazione da un'origine dati diversa: prima interroga la tabella di destinazione per ottenere la data più recente, quindi interroga la tabella di origine per i dati più recenti e li inserisce nella tabella di destinazione.

Fase 1: Creare la macchina a stati e fornire risorse

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`Keep data up to date`** nella casella di ricerca, quindi scegli Mantieni i dati aggiornati dai risultati di ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Questo progetto di esempio utilizza le seguenti risorse:

   - Un bucket Amazon S3
   - Amazon Athenainterrogazioni
   - Qualsiasi chiamata AWS Glue Data Catalog
   - Una macchina a stati AWS Step Functions
   - Ruoli AWS Identity and Access Management (IAM) correlati

L'immagine seguente mostra il grafico del flusso di lavoro per il progetto di esempio Keep data update:

5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

     In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

     > **Important**
     >
     > Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

- Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

  > **Tip**
  >
  > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

  Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

  La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Mentre le risorse vengono distribuite, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

  Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

  > **Important**
  >
  > Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.
2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

   1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

> **Note**
>
> Step Functions consente di creare nomi per macchine a stati, esecuzioni, attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

   Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.
3. Selezionare Start execution (Avvia esecuzione).
4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

   Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Esempio di codice della macchina a stati

La macchina a stati di questo progetto di esempio si integra con Amazon S3 AWS Glue e Amazon Athena passando i parametri direttamente a tali risorse.

Sfoglia questo esempio di macchina a stati per vedere come Step Functions controlla Amazon S3 e Amazon Athena connettendosi all'Amazon Resource Name (ARN) sul campo e passando `Resource Parameters` all'API del servizio. AWS Glue

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consulta. Utilizzo AWS Step Functions con altri servizi

```
{
    "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
    }
  }
}
```

Esempio IAM

Questo esempio di policy AWS Identity and Access Management (IAM) generato dal progetto di esempio include il minimo privilegio necessario per eseguire la macchina a stati e le risorse correlate. Ti consigliamo di includere solo le autorizzazioni necessarie nelle tue policy IAM.

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

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Gestisci un cluster Amazon EKS

Questo progetto di esempio dimostra come utilizzare Step Functions e Amazon Elastic Kubernetes Service per creare un cluster Amazon EKS con un gruppo di nodi, eseguire un job su Amazon EKS, quindi esaminare l'output. Al termine, rimuove i gruppi di nodi e il cluster Amazon EKS.

Per ulteriori informazioni sulle integrazioni dei servizi Step Functions e Step Functions, consulta quanto segue:

- Utilizzo AWS Step Functions con altri servizi
- Chiama Amazon EKS con Step Functions

> **Note**
>
> Questo progetto di esempio potrebbe comportare costi.
> Per i nuovi AWS utenti, è disponibile un piano di utilizzo gratuito. In questo piano, i servizi sono gratuiti al di sotto di un determinato livello di utilizzo. Per ulteriori informazioni sui AWS costi e sul piano gratuito, consulta i prezzi di Amazon EKS.

Fase 1: Creare la macchina a stati e fornire risorse

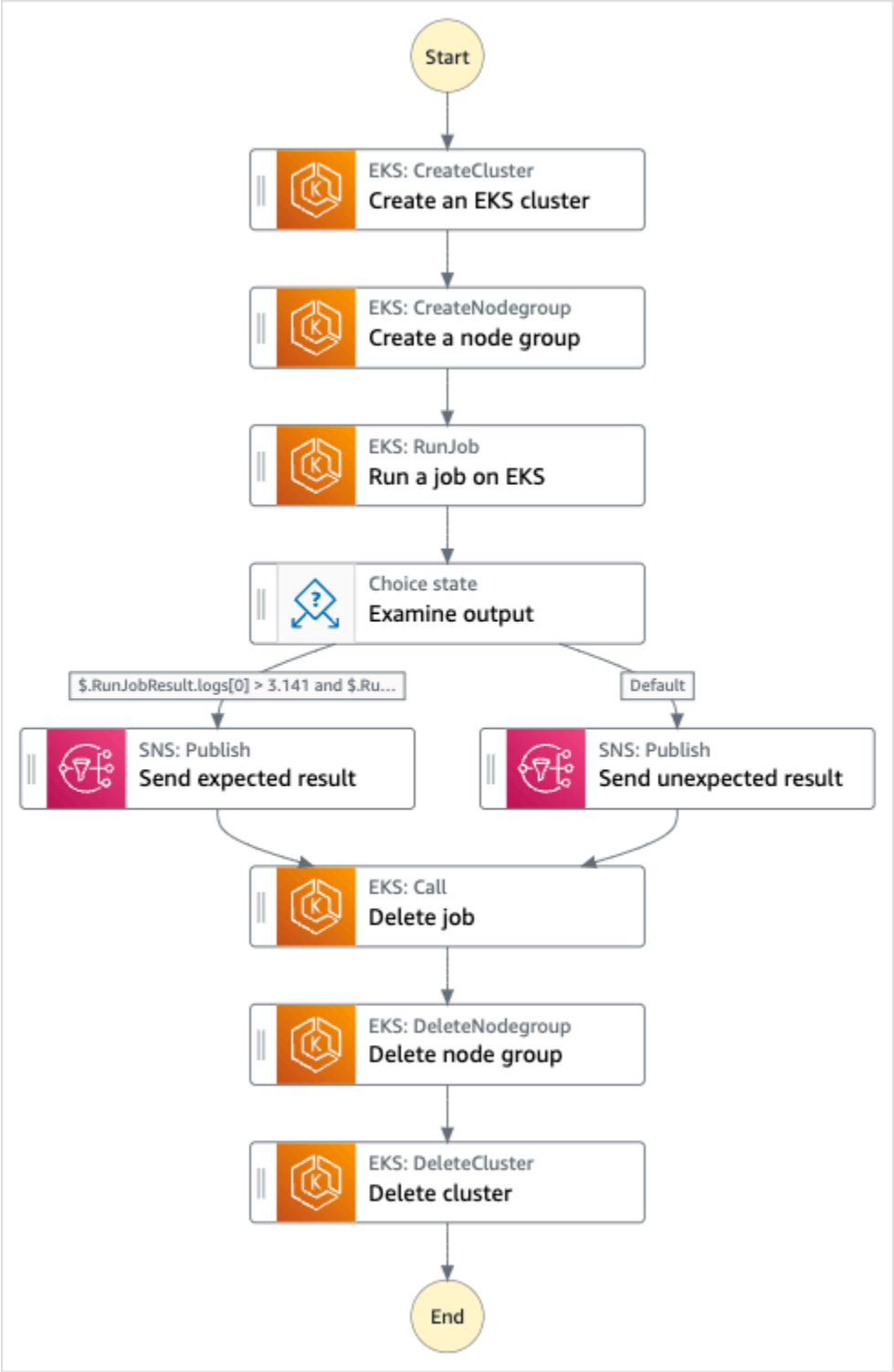
1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`Manage an EKS cluster`** nella casella di ricerca, quindi scegli Gestisci un cluster EKS dai risultati della ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Questo progetto di esempio utilizza le seguenti risorse:

   - Un cluster Amazon Elastic Kubernetes Service
   - Un argomento Amazon SNS
   - Una macchina a stati AWS Step Functions

- Ruoli AWS Identity and Access Management (IAM) correlati

L'immagine seguente mostra il grafico del flusso di lavoro per il progetto di esempio Manage an EKS cluster:



5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

     In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

     > **Important**
     >
     > Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

   - Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

     > **Tip**
     >
     > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

     Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

     La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Durante la distribuzione delle risorse, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

     Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

> **Important**
>
> Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.
2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

    1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

        > **Note**
        >
        > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

    2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

        Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

        > **Note**
        >
        > Se il progetto demo che hai distribuito contiene dati di input di esecuzione precompilati, usa quell'input per eseguire la macchina a stati.

    3. Selezionare Start execution (Avvia esecuzione).
    4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile

esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Esempio di codice della macchina a stati

La macchina a stati di questo progetto di esempio si integra con Amazon EKS creando un cluster e un gruppo di nodi Amazon EKS e utilizza un argomento SNS per restituire risultati.

Sfoglia questo esempio di macchina a stati per scoprire come Step Functions gestisce i cluster e i gruppi di nodi Amazon EKS.

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consultaUtilizzo AWS Step Functions con altri servizi.

```
{
  "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
    }
  }
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Esempio IAM

Questi esempi di policy AWS Identity and Access Management (IAM) generati dal progetto di esempio includono il privilegio minimo necessario per eseguire la macchina a stati e le risorse correlate. Ti consigliamo di includere solo le autorizzazioni necessarie nelle tue policy IAM.

```
{
    "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"
            ]
        }
    ]
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Effettuare una chiamata a API Gateway

Questo progetto di esempio dimostra come utilizzare Step Functions per effettuare una chiamata ad API Gateway e verifica se la chiamata è riuscita.

Per ulteriori informazioni sulle integrazioni dei servizi API Gateway e Step Functions, consulta quanto segue:

- Utilizzo AWS Step Functions con altri servizi
- Chiama API Gateway con Step Functions

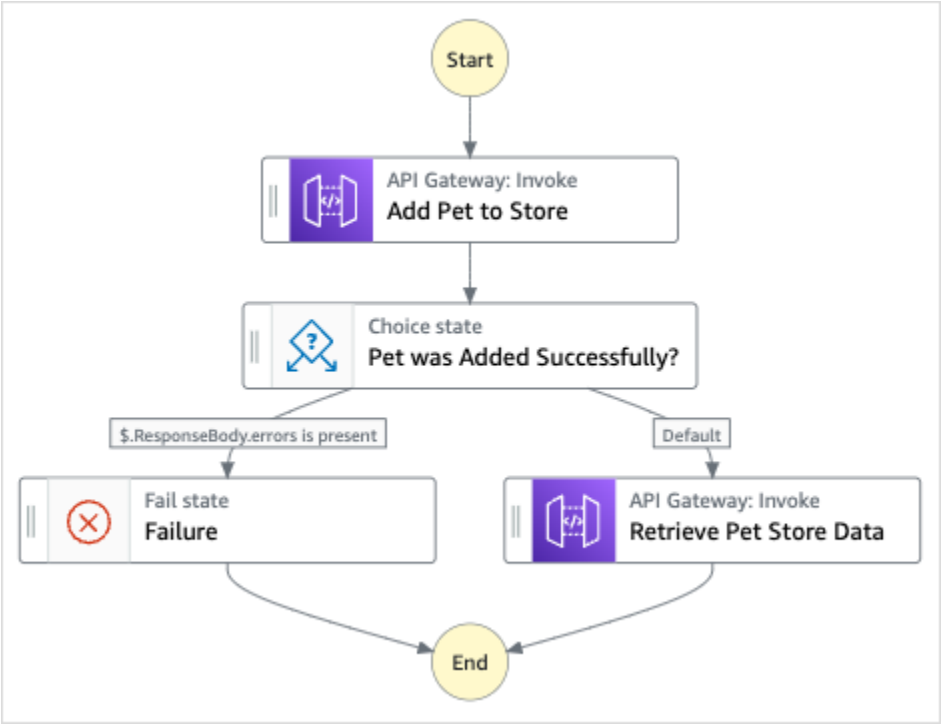
Fase 1: Creare la macchina a stati e fornire risorse

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`Make a call to API Gateway`** nella casella di ricerca, quindi scegli Effettua una chiamata a API Gateway tra i risultati della ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Questo progetto di esempio utilizza le seguenti risorse:

   - Un'API Amazon API Gateway REST che viene chiamata dalla macchina a stati.
   - Una macchina a stati AWS Step Functions
   - Ruoli AWS Identity and Access Management (IAM) correlati

   L'immagine seguente mostra il grafico del flusso di lavoro per il progetto Make a call to API Gateway sample:



5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

     In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

     > **Important**
     >
     > Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

   - Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

> **Tip**
>
> Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Durante la distribuzione delle risorse, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

> **Important**
>
> Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.
2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

    1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

        > **Note**
        >
        > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano

con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

   Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

   > **Note**
   >
   > Se il progetto demo che hai distribuito contiene dati di input di esecuzione precompilati, usa quell'input per eseguire la macchina a stati.

3. Selezionare Start execution (Avvia esecuzione).
4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

   Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Esempio di codice della macchina a stati

La macchina a stati di questo progetto di esempio si integra con API Gateway chiamando l'API REST di API Gateway e passando tutti i parametri necessari.

Sfoglia questo esempio di macchina a stati per vedere come Step Functions interagisce con API Gateway.

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consultaUtilizzo AWS Step Functions con altri servizi.

```
{
  "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
    }
  }
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Esempio IAM

Questi esempi di policy AWS Identity and Access Management (IAM) generati dal progetto di esempio includono il privilegio minimo necessario per eseguire la macchina a stati e le risorse correlate. Ti consigliamo di includere solo le autorizzazioni necessarie nelle tue policy IAM.

```
{
    "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"
        }
    ]
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Chiama un microservizio in esecuzione su Fargate utilizzando l'integrazione API Gateway

Questo progetto di esempio dimostra come utilizzare Step Functions per effettuare una chiamata ad API Gateway per interagire con un servizio attivo e anche per verificare se la chiamata è riuscita. AWS Fargate

Per ulteriori informazioni sulle integrazioni dei servizi API Gateway e Step Functions, consulta quanto segue:

- Utilizzo AWS Step Functions con altri servizi
- Chiama API Gateway con Step Functions

> **Note**
>
> Questo progetto di esempio potrebbe comportare costi.
> Per i nuovi AWS utenti, è disponibile un livello di utilizzo gratuito. In questo piano, i servizi sono gratuiti al di sotto di un determinato livello di utilizzo. Per ulteriori informazioni sui AWS costi e sul piano gratuito, consulta la sezione Prezzi.

Fase 1: Creazione della macchina a stati e fornitura delle risorse

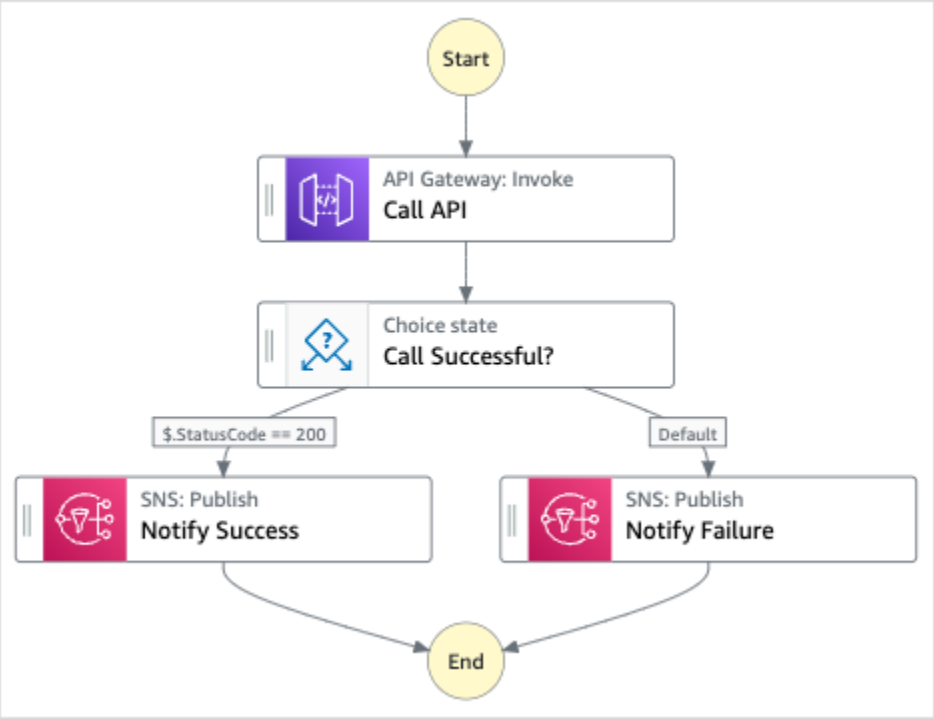
1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`Call a microservice with API Gateway`** nella casella di ricerca, quindi scegli Chiama un microservizio con API Gateway tra i risultati della ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Questo progetto di esempio utilizza le seguenti risorse:

   - Un'API Amazon API Gateway HTTP che viene chiamata dalla macchina a stati.
   - Un Amazon API Gateway Amazon VPC link.

- Un Amazon Virtual Private Cloud.
- Un Application Load Balancer.
- Un Fargate grappolo.
- Un argomento Amazon SNS
- Una macchina a AWS Step Functions stati
- Ruoli correlati AWS Identity and Access Management (IAM)
- Diversi servizi aggiuntivi necessari per consentire a queste risorse di funzionare insieme.

L'immagine seguente mostra il grafico del flusso di lavoro per il progetto Call a microservice with API Gateway sample:



5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

     In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione

Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

> **Important**
>
> Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

- Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

  > **Tip**
  >
  > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

  Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

  La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Mentre le risorse vengono distribuite, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

  Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

  > **Important**
  >
  > Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.

2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

   1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

      > **Note**
      >
      > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

   2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

      Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

      > **Note**
      >
      > Se il progetto demo che hai distribuito contiene dati di input di esecuzione precompilati, usa quell'input per eseguire la macchina a stati.

   3. Selezionare Start execution (Avvia esecuzione).
   4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

      Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Esempio di codice della macchina a stati

La macchina a stati di questo progetto di esempio si integra con API Gateway chiamando un'API HTTP API Gateway connessa a un servizio su Fargate. È ospitato su una sottorete privata e vi si accede tramite un sistema di bilanciamento del carico delle applicazioni privato.

Sfoglia questo esempio di macchina a stati per vedere come Step Functions interagisce con API Gateway e restituisce risultati.

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consultaUtilizzo AWS Step Functions con altri servizi.

```
{
  "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
    }
  }
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Esempio IAM

Questi esempi di policy AWS Identity and Access Management (IAM) generati dal progetto di esempio includono il privilegio minimo necessario per eseguire la macchina a stati e le risorse correlate. Ti consigliamo di includere solo le autorizzazioni necessarie nelle tue policy IAM.

```
{
    "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"
        }
```

```
    ]
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Invia un evento personalizzato a EventBridge

Questo progetto di esempio dimostra come utilizzare Step Functions per inviare un evento personalizzato a un bus di eventi che corrisponde a una regola con più destinazioni (Amazon EventBridge, Amazon Simple Notification Service AWS Lambda, Amazon Simple Queue Service).

Per ulteriori informazioni sulle integrazioni dei servizi Step Functions e Step Functions, consulta quanto segue:

- Utilizzo AWS Step Functions con altri servizi
- Chiama EventBridge con Step Functions

> **Note**
>
> Questo progetto di esempio potrebbe comportare costi.
> Per i nuovi AWS utenti, è disponibile un piano di utilizzo gratuito. In questo piano, i servizi sono gratuiti al di sotto di un determinato livello di utilizzo. Per ulteriori informazioni sui AWS costi e sul piano gratuito, consulta la sezione EventBridge Prezzi.

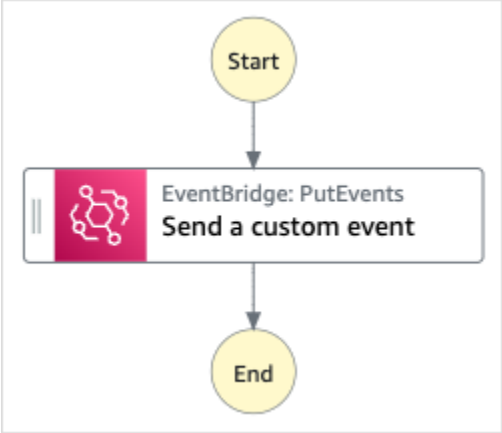
Fase 1: Creazione della macchina a stati e fornitura delle risorse

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`Send a custom event to EventBridge`** nella casella di ricerca, quindi scegli Invia un evento personalizzato a EventBridge tra i risultati della ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

Questo progetto di esempio utilizza le seguenti risorse:

- Un evento Amazon EventBridge
- Un argomento Amazon SNS
- Una coda Amazon SQS
- Una funzione Lambda
- Una macchina AWS Step Functions statale
- Ruoli correlati AWS Identity and Access Management (IAM)

L'immagine seguente mostra il grafico del flusso di lavoro per il progetto di EventBridge esempio Invia un evento personalizzato:



5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

     In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

> **Important**
>
> Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

- Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

  > **Tip**
  >
  > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

  Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

  La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Mentre le risorse vengono distribuite, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

  Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

  > **Important**
  >
  > Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.
2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

   > **Note**
   >
   > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

   Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

   > **Note**
   >
   > Se il progetto demo che hai distribuito contiene dati di input di esecuzione precompilati, usa quell'input per eseguire la macchina a stati.

3. Selezionare Start execution (Avvia esecuzione).
4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

   Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Esempio di codice della macchina a stati

La macchina a stati di questo progetto di esempio si integra EventBridge inviando un evento personalizzato a un bus di EventBridge eventi. L'evento inviato al bus degli eventi corrisponde a una

EventBridge regola che attiva una funzione Lambda che invia messaggi a un argomento Amazon SNS e a una coda Amazon SQS.

Sfoglia questo esempio di macchina a stati per vedere come funziona Step Functions EventBridge.

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consultaUtilizzo AWS Step Functions con altri servizi.

```
{
  "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
    }
  }
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Esempio IAM

Questi esempi di policy AWS Identity and Access Management (IAM) generati dal progetto di esempio includono il privilegio minimo necessario per eseguire la macchina a stati e le risorse correlate. Ti consigliamo di includere solo le autorizzazioni necessarie nelle tue policy IAM.

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

```
    "Statement": [
        {
            "Action": [
                "events:PutEvents"
            ],
            "Resource": [
                "arn:aws:events:us-east-1:1234567890:event-bus/stepfunctions-
sampleproject-eventbus"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Richiama flussi di lavoro Synchronous Express

Questo progetto di esempio dimostra come richiamare i flussi di lavoro Synchronous Express tramite Amazon API Gateway per gestire un database di dipendenti.

In questo progetto, Step Functions utilizza gli endpoint API Gateway per avviare i flussi di lavoro Step Functions Synchronous Express. Questi utilizzano quindi DynamoDB per cercare, aggiungere e rimuovere dipendenti in un database dei dipendenti.

Per ulteriori informazioni sui flussi di lavoro Step Functions Synchronous Express, vedere. Flussi di lavoro Express sincroni e asincroni

> **Note**
>
> Questo progetto di esempio potrebbe comportare costi.
> Per i nuovi AWS utenti, è disponibile un livello di utilizzo gratuito. In questo piano, i servizi sono gratuiti al di sotto di un determinato livello di utilizzo. Per ulteriori informazioni sui AWS costi e sul piano gratuito, consulta i prezzi di Step Functions.

Fase 1: Creazione della macchina a stati e fornitura delle risorse

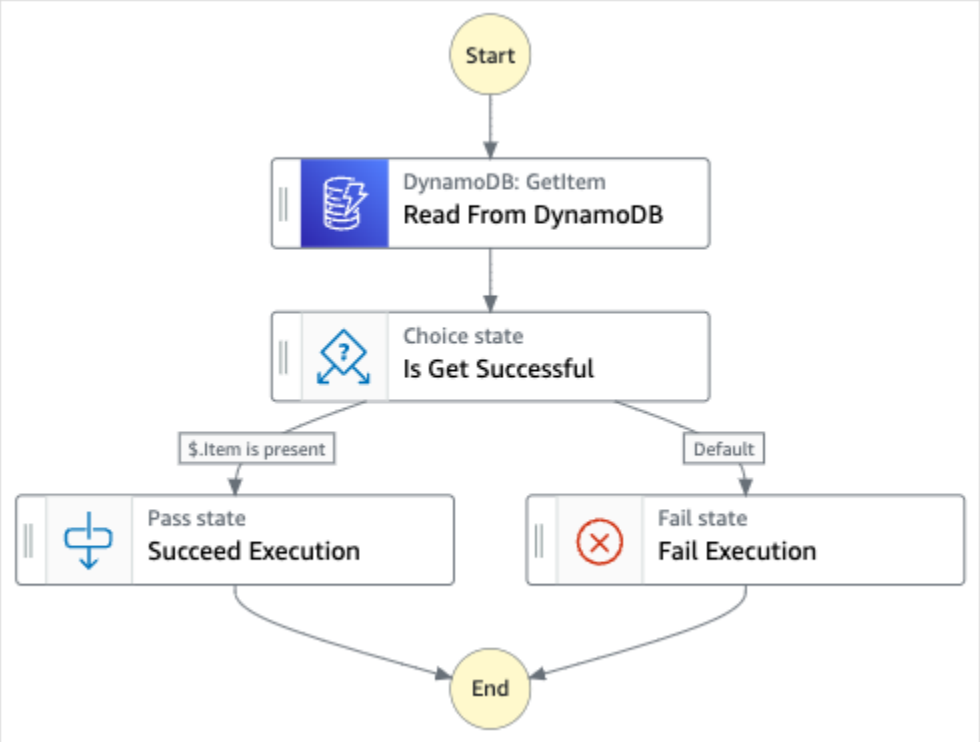
1. Apri la console Step Functions e scegli Crea macchina a stati.

2. Digita **`Invoke Synchronous Express Workflows through API Gateway`** nella casella di ricerca, quindi scegli Invoke Synchronous Express Workflows API Gateway tra i risultati della ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Questo progetto di esempio utilizza le seguenti risorse:

   - Un'API Amazon API Gateway HTTPS richiamata da una macchina a stati.
   - Una tabella Amazon DynamoDB.
   - Tre macchine a AWS Step Functions stati.
   - Ruoli correlati AWS Identity and Access Management (IAM).

   L'immagine seguente mostra il grafico del flusso di lavoro per Invoke Synchronous Express Workflows tramite API Gateway un progetto di esempio:



5. Scegli Usa modello per continuare con la selezione.

6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

     In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

     > **Important**
     >
     > Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

   - Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

     > **Tip**
     >
     > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

     Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

     La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Mentre le risorse vengono distribuite, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

     Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

> **Important**
>
> Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.
2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

    1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

        > **Note**
        >
        > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

    2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

        Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

        > **Note**
        >
        > Se il progetto demo che hai distribuito contiene dati di input di esecuzione precompilati, usa quell'input per eseguire la macchina a stati.

    3. Selezionare Start execution (Avvia esecuzione).
    4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile

esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Esempio di codice della macchina a stati

La macchina a stati di questo progetto di esempio si integra con API Gateway e DynamoDB utilizzando API Gateway per richiamare un Synchronous Express Workflow, che quindi aggiorna o legge dal database dei dipendenti utilizzando DynamoDB.

Sfogliate questo esempio di macchina a stati per vedere come Step Functions legge da DynamoDB per recuperare le informazioni sui dipendenti.

Per ulteriori informazioni su come richiamare Step Functions utilizzando API Gateway, vedere quanto segue.

- Chiama API Gateway con Step Functions
- Come richiamare un gateway privato nella API Gateway Developer Guide.

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consultaUtilizzo AWS Step Functions con altri servizi.

```
{
  "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"
    }
  }
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vedere[Politiche IAM per servizi integrati](#).

Esempi IAM

Questi esempi di policy AWS Identity and Access Management (IAM) generati dal progetto di esempio includono il minimo privilegio necessario per eseguire la macchina a stati e le risorse correlate. Ti consigliamo di includere solo le autorizzazioni necessarie nelle tue policy IAM.

```
{
    "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"
            ]
        }
    ]
```

```
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Esegui flussi di lavoro ETL/ELT utilizzando Amazon Redshift (Lambda, Amazon Redshift Data API)

Questo progetto di esempio dimostra come utilizzare Step Functions e l'Amazon Redshift Data API per eseguire un flusso di lavoro ETL/ELT che carica i dati nel data warehouse Amazon Redshift.

In questo progetto, Step Functions utilizza una AWS Lambda funzione e l'Amazon Redshift Data API per creare gli oggetti di database richiesti e generare un set di dati di esempio, quindi esegue due processi in parallelo che eseguono il caricamento di tabelle delle dimensioni, seguiti da una tabella dei fatti. Una volta terminati correttamente i processi di caricamento di entrambe le dimensioni, Step Functions esegue il processo di caricamento per la tabella dei fatti, esegue il processo di convalida, quindi mette in pausa il cluster Amazon Redshift.

> **Note**
>
> Puoi modificare la logica ETL per ricevere dati da altre fonti come Amazon S3, che può utilizzare il comando COPY per copiare dati da Amazon S3 a una tabella Amazon Redshift.

Per ulteriori informazioni sulle integrazioni dei servizi Amazon Redshift e Step Functions, consulta quanto segue:

- Utilizzo AWS Step Functions con altri servizi
- Utilizzo dell'API dati Amazon Redshift
- Servizio Amazon Redshift Data API
- Creazione di una macchina a stati Step Functions che utilizza Lambda
- Politiche IAM per:
  - Politiche IAM per AWS Lambda
  - Autorizzazione dell'accesso all'API dei dati di Amazon Redshift

> **Note**
>
> Questo progetto di esempio potrebbe comportare costi.
> Per i nuovi AWS utenti, è disponibile un piano di utilizzo gratuito. In questo piano, i servizi sono gratuiti al di sotto di un determinato livello di utilizzo. Per ulteriori informazioni sui AWS costi e sul piano gratuito, consulta AWS Step Functions i prezzi.

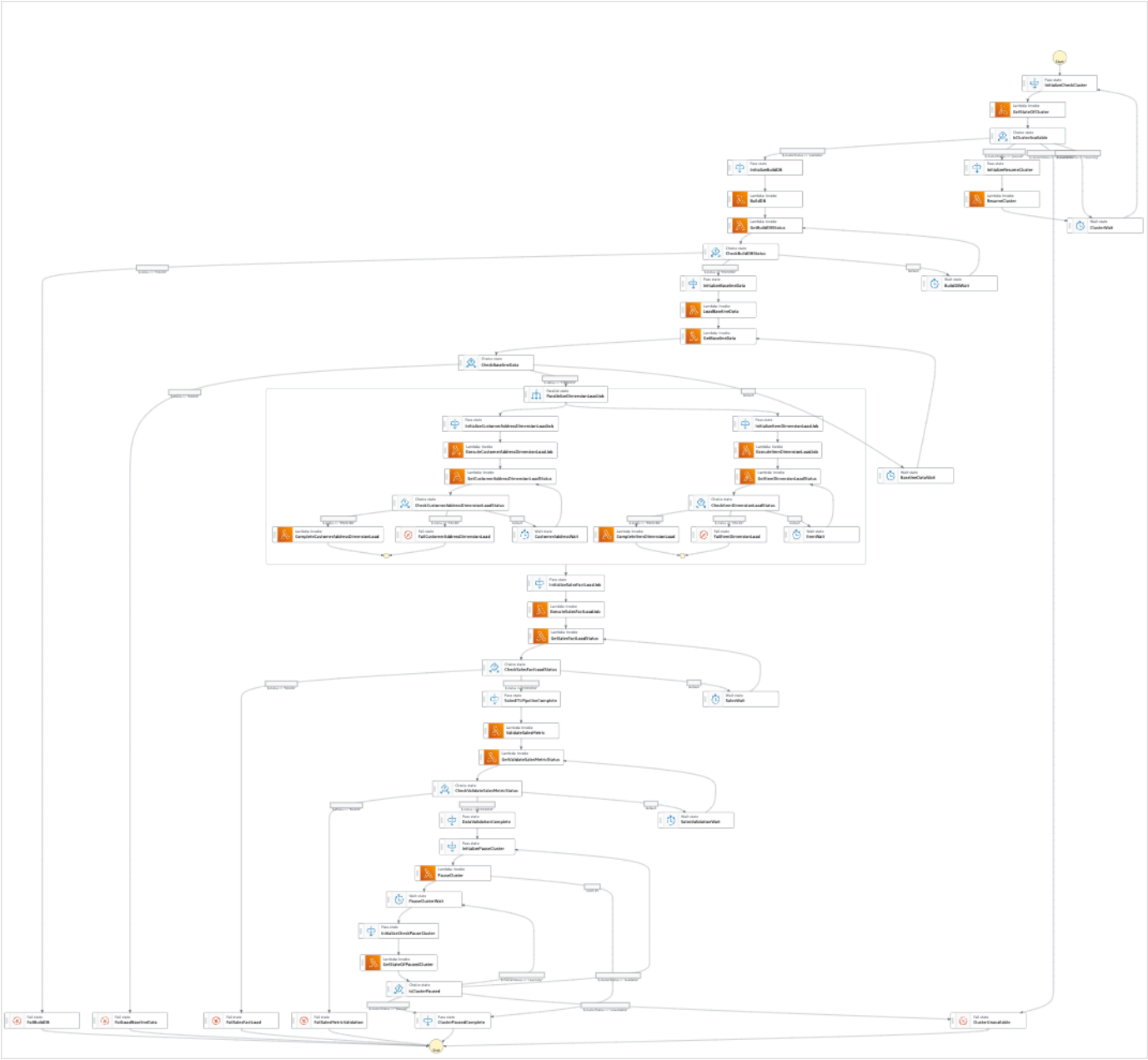
Fase 1: Creazione della macchina a stati e fornitura delle risorse

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`ETL job in Amazon Redshift`** nella casella di ricerca, quindi scegli ETL job in Amazon Redshift dai risultati di ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Questo progetto di esempio utilizza le seguenti risorse:

   - Un cluster Amazon Redshift
   - Due funzioni Lambda
   - Uno Amazon Redshift schema
   - Cinque Amazon Redshift tavoli
   - Una macchina AWS Step Functions statale
   - Ruoli correlati AWS Identity and Access Management (IAM).

   L'immagine seguente mostra il grafico del flusso di lavoro per il job ETL nel progetto Amazon Redshift di esempio:



5. Scegliete Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

     In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione

Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

> **Important**
>
> Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

- Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

  > **Tip**
  >
  > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

  Quando sei pronto, scegli Implementa ed esegui per distribuire il progetto di esempio e creare le risorse.

  La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Mentre le risorse vengono distribuite, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

  Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

  > **Important**
  >
  > Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.

2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

   1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

      > **Note**
      >
      > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

   2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

      Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

      > **Note**
      >
      > Se il progetto demo che hai distribuito contiene dati di input di esecuzione precompilati, usa quell'input per eseguire la macchina a stati.

   3. Selezionare Start execution (Avvia esecuzione).
   4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

      Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Esempio di codice della macchina a stati

La macchina a stati di questo progetto di esempio si integra AWS Lambda trasmettendo la logica ETL InputPath direttamente a tali risorse e viene eseguita in modo asincrono utilizzando l'API Amazon Redshift Data.

Sfoglia questo esempio di macchina a stati per vedere come Step Functions controlla AWS Lambda e l'Amazon Redshift Data API.

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consultaUtilizzo AWS Step Functions con altri servizi.

```
{
"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
    }
  }
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Esempio IAM

Questi esempi di policy AWS Identity and Access Management (IAM) generati dal progetto di esempio includono il privilegio minimo necessario per eseguire la macchina a stati e le risorse correlate. Ti consigliamo di includere solo le autorizzazioni necessarie nelle tue policy IAM.

```
{
    "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"
        }
    ]
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Utilizzo Step Functions e AWS Batch con gestione degli errori

Questo progetto di esempio dimostra come eseguire un AWS Batch processo utilizzando una Step Functions macchina a stati con funzionalità di gestione degli errori.

Step Functions, in questa occasione, si avvale di una macchina a stati per invocare il processo in AWS Batch in modo sincrono. Attende quindi che il processo abbia esito positivo o negativo, riprova e rileva gli errori quando un processo fallisce, quindi invia un Amazon SNS argomento con un messaggio indicante se il processo è riuscito o meno.

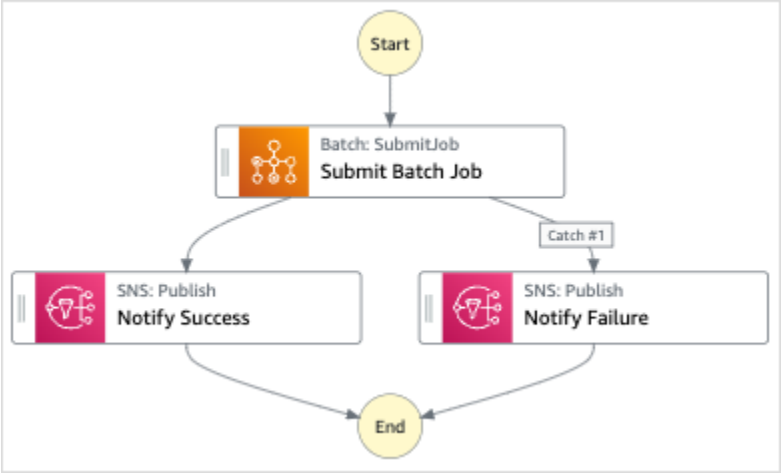
Fase 1: Creare la macchina a stati e fornire risorse

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`Manage a batch job`** nella casella di ricerca, quindi scegli Gestisci un processo in batch dai risultati della ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Questo progetto di esempio utilizza le seguenti risorse:

   - Un lavoro AWS Batch
   - Un argomento Amazon SNS
   - Una macchina AWS Step Functions statale
   - Ruoli correlati AWS Identity and Access Management (IAM)

L'immagine seguente mostra il grafico del flusso di lavoro per il progetto di esempio Manage a batch job:



5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

     In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

     > **Important**
     >
     > Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

   - Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

> **Tip**
>
> Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Mentre le risorse vengono distribuite, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

> **Important**
>
> Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.
2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

    1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

        > **Note**
        >
        > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano

con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

   Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

   > **Note**
   >
   > Se il progetto demo che hai distribuito contiene dati di input di esecuzione precompilati, usa quell'input per eseguire la macchina a stati.

3. Selezionare Start execution (Avvia esecuzione).
4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

   Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Esempio di codice della macchina a stati

La macchina a stati di questo progetto di esempio si integra con AWS Batch Amazon SNS passando i parametri direttamente a tali risorse.

Sfoglia questo esempio di macchina a stati per vedere come Step Functions controlla AWS Batch Amazon SNS connettendosi all'Amazon Resource Name (ARN) sul `Resource` campo e passando `Parameters` all'API del servizio.

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consulta. Utilizzo AWS Step Functions con altri servizi

```
{
```

```
  "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
    }
  }
}
```

Esempio IAM

Questo esempio di policy AWS Identity and Access Management (IAM) generato dal progetto di esempio include il minimo privilegio necessario per eseguire la macchina a stati e le risorse correlate. Ti consigliamo di includere solo le autorizzazioni necessarie nelle tue policy IAM.

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"
        }
    ]
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vedereIAM Policies for Integrated Services.

Trova un AWS Batch lavoro

Questo progetto di esempio dimostra come utilizzare Eseguire la mappatura lo stato di Step Functions per eseguire AWS Batch lavori a tutto volume.

In questo progetto, Step Functions utilizza una macchina a stati per richiamare una funzione Lambda per eseguire una semplice preelaborazione, quindi richiama AWS Batch più lavori in parallelo utilizzando lo stato. Eseguire la mappatura

Fase 1: Creare la macchina a stati e fornire risorse

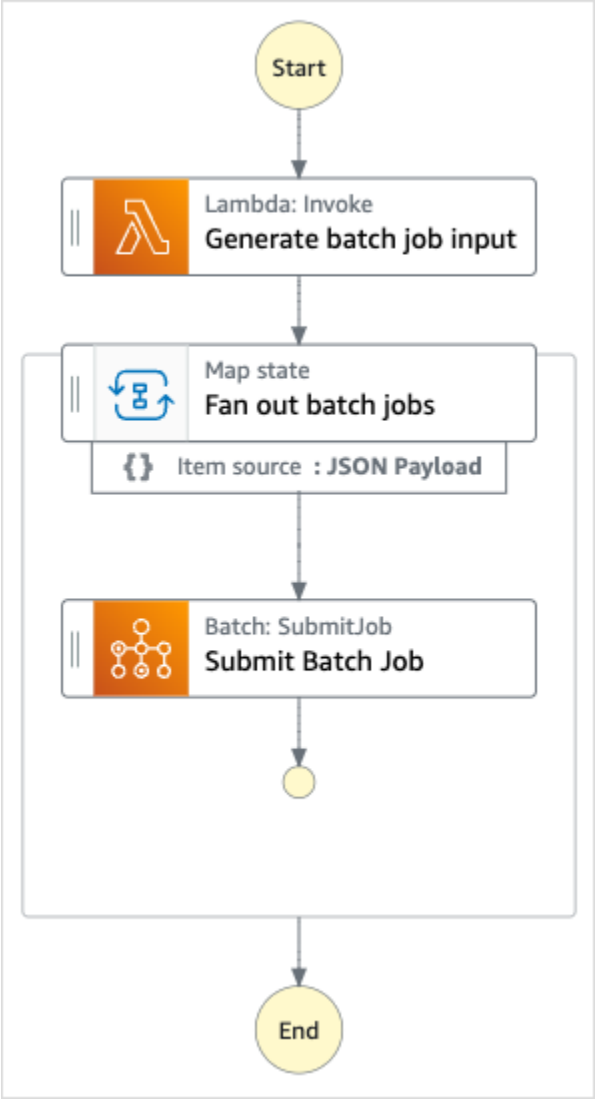
1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`Fan out a batch job`** nella casella di ricerca, quindi scegli Esegui un lavoro in batch dai risultati di ricerca che vengono restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

   Questo progetto di esempio utilizza le seguenti risorse:

   - Una funzione Lambda

- Una coda di AWS Batch lavoro
- Una macchina AWS Step Functions statale
- Ruoli correlati AWS Identity and Access Management (IAM)

L'immagine seguente mostra il grafico del flusso di lavoro per il progetto di esempio Fan out a batch job:



5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

> **Important**
>
> Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

- Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

  > **Tip**
  >
  > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

  Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

  La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Durante la distribuzione delle risorse, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

  Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

  > **Important**
  >
  > Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.
2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

   1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

      > **Note**
      >
      > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

   2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

      Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

      > **Note**
      >
      > Se il progetto demo che hai distribuito contiene dati di input di esecuzione precompilati, usa quell'input per eseguire la macchina a stati.

   3. Selezionare Start execution (Avvia esecuzione).
   4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

      Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Esempio di codice della macchina a stati

La macchina a stati di questo progetto di esempio si integra con AWS Batch Amazon SNS passando i parametri direttamente a tali risorse.

Sfoglia questo esempio di macchina a stati per vedere come Step Functions controlla AWS Batch Amazon SNS connettendosi all'Amazon Resource Name (ARN) sul `Resource` campo e passando `Parameters` all'API del servizio.

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consulta. Utilizzo AWS Step Functions con altri servizi

```
{
  "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
          }
        }
      }
    }
  }
}
```

Esempio IAM

Questi esempi di policy AWS Identity and Access Management (IAM) generati dal progetto di esempio includono il minimo privilegio necessario per eseguire la macchina a stati e le risorse correlate. Ti consigliamo di includere solo le autorizzazioni necessarie nelle tue policy IAM.

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"
        }
    ]
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vedereIAM Politiche per servizi integrati.

AWS Batch con Lambda

Questo progetto di esempio dimostra come utilizzare Step Functions per preelaborare i dati con AWS Lambda funzioni e quindi AWS Batch orchestrare i lavori.

In questo progetto, Step Functions utilizza una macchina a stati per richiamare una funzione Lambda per eseguire una semplice preelaborazione prima dell'invio di AWS Batch un lavoro. È possibile richiamare più job a seconda del risultato o del successo di quello precedente.

Fase 1: Creazione della macchina statale e fornitura di risorse

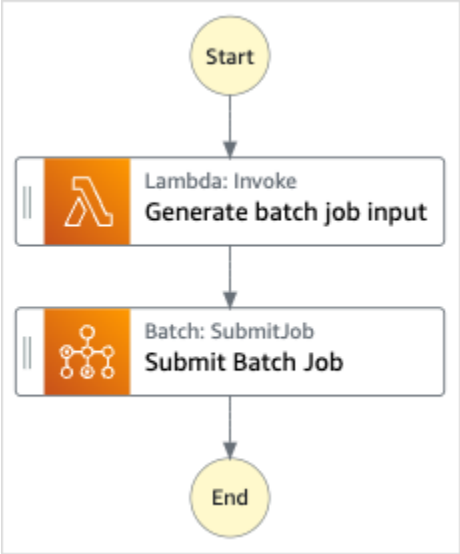
1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`Batch job with Lambda`** nella casella di ricerca, quindi scegli Batch job with Lambda dai risultati di ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul

tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

Questo progetto di esempio utilizza le seguenti risorse:

- Una funzione Lambda
- Un processo in AWS Batch
- Una macchina a AWS Step Functions stati
- Ruoli correlati AWS Identity and Access Management (IAM)

L'immagine seguente mostra il grafico del flusso di lavoro per il processo Batch con progetto Lambda di esempio:



5. Scegliete Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

   - Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

     In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step

Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

> **Important**
>
> Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

- Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

  > **Tip**
  >
  > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

  Quando sei pronto, scegli Implementa ed esegui per distribuire il progetto di esempio e creare le risorse.

  La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Mentre le risorse vengono distribuite, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

  Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

  > **Important**
  >
  > Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.
2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.

3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

   1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

      > **Note**
      >
      > Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

   2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

      Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

      > **Note**
      >
      > Se il progetto demo che hai distribuito contiene dati di input di esecuzione precompilati, usa quell'input per eseguire la macchina a stati.

   3. Selezionare Start execution (Avvia esecuzione).
   4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

      Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vederePagina dei dettagli di esecuzione: panoramica dell'interfaccia.

Esempio di codice della macchina a stati

La macchina a stati di questo progetto di esempio si integra con AWS Batch Amazon SNS passando i parametri direttamente a tali risorse.

Sfoglia questo esempio di macchina a stati per vedere come Step Functions controlla AWS Batch Amazon SNS connettendosi all'Amazon Resource Name (ARN) sul `Resource` campo e passando `Parameters` all'API del servizio.

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consulta. Utilizzo AWS Step Functions con altri servizi

```
{
  "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
    }
  }
}
```

Esempio IAM

Questi esempi di policy AWS Identity and Access Management (IAM) generati dal progetto di esempio includono il minimo privilegio necessario per eseguire la macchina a stati e le risorse correlate. Ti consigliamo di includere solo le autorizzazioni necessarie nelle tue policy IAM.

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"
        }
    ]
}
```

Per informazioni su come configurare IAM quando si utilizza Step Functions con altri AWS servizi, vederePolitiche IAM per servizi integrati.

Esegui il concatenamento dei prompt AI con Amazon Bedrock

Questo progetto di esempio dimostra come è possibile eseguire l'integrazione con Amazon Bedrock per eseguire il prompt-chaining basato sull'intelligenza artificiale. Questo progetto di esempio mostra come è possibile creare chatbot di alta qualità utilizzando. Amazon Bedrock Il progetto riunisce alcuni prompt e li risolve nella sequenza in cui vengono forniti. Il concatenamento di questi prompt aumenta la capacità del modello linguistico utilizzato di fornire una risposta altamente curata.

Questo progetto di esempio crea la macchina a stati, le AWS risorse di supporto e configura le relative autorizzazioni IAM. Esplora questo progetto di esempio per scoprire come utilizzare l'integrazione Amazon Bedrock ottimizzata dei servizi con le macchine a Step Functions stati o usalo come punto di partenza per i tuoi progetti.

Argomenti

- Modello AWS CloudFormation e risorse aggiuntive
- Prerequisiti
- Fase 1: Creare la macchina a stati e fornire risorse

- Fase 2: Eseguire la macchina a stati

Modello AWS CloudFormation e risorse aggiuntive

Utilizzate un CloudFormation modello per distribuire questo progetto di esempio. Questo modello crea le seguenti risorse nel tuo Account AWS:

- Una macchina a Step Functions stati.
- Ruolo di esecuzione per la macchina a stati. Questo ruolo concede le autorizzazioni necessarie alla macchina a stati per accedere ad altre Servizi AWS risorse come l'azione Amazon Bedrock InvokeModel.

Prerequisiti

Questo progetto di esempio utilizza il modello di linguaggio di grandi dimensioni (LLM) di Cohere Command. Per eseguire correttamente questo progetto di esempio, è necessario aggiungere l'accesso a questo LLM dalla console. Amazon Bedrock Per aggiungere l'accesso al modello, procedi come segue:

1. Apri la console Amazon Bedrock.
2. Nel pannello di navigazione, scegli Model access.
3. Scegli Gestisci l'accesso al modello.
4. Seleziona la casella di controllo accanto a Cohere.
5. Scegli Richiedi accesso. Lo stato di accesso per il modello Cohere viene visualizzato come Accesso concesso.

Fase 1: Creare la macchina a stati e fornire risorse

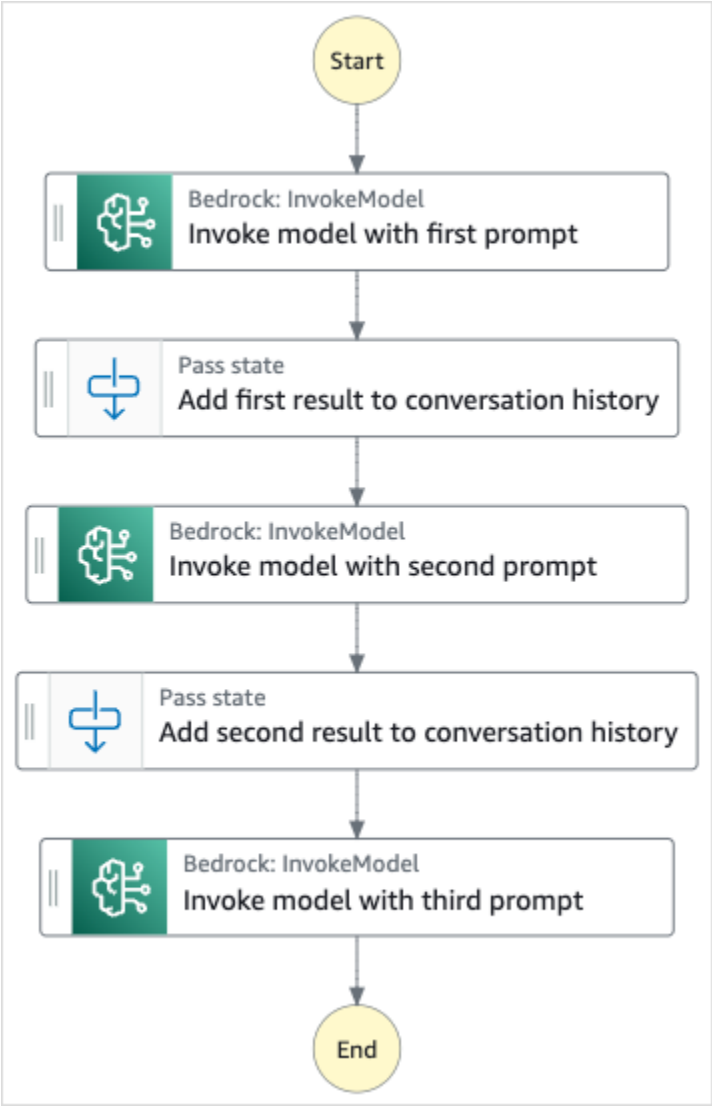
1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Digita **`bedrock`** nella casella di ricerca, quindi scegli Perform AI prompt-chaining con tra i risultati Bedrock della ricerca restituiti.
3. Seleziona Successivo per continuare.
4. Step Functions elenca quelli Servizi AWS utilizzati nel progetto di esempio selezionato. Mostra anche un grafico del flusso di lavoro per il progetto di esempio. Implementa questo progetto sul

tuo Account AWS o usalo come punto di partenza per creare i tuoi progetti. In base a come vuoi procedere, scegli Esegui una demo o Sviluppa su di essa.

Questo progetto di esempio utilizza le seguenti risorse:

- Una macchina a AWS Step Functions stati
- Ruoli correlati AWS Identity and Access Management (IAM)

L'immagine seguente mostra il grafico del flusso di lavoro per il prompt-chaining Perform AI con il progetto di esempio: Bedrock



5. Scegli Usa modello per continuare con la selezione.
6. Esegui una di queste operazioni:

- Se hai selezionato Build on it, Step Functions crea il prototipo del flusso di lavoro per il progetto di esempio selezionato. Step Functions non distribuisce le risorse elencate nella definizione del flusso di lavoro.

  In Workflow StudioModalità di progettazione, trascina gli stati da e continua Stati del browser a creare il tuo prototipo di flusso di lavoro. Oppure passa a Modalità codice quello che fornisce un editor di codice integrato simile a VS Code per aggiornare la definizione Amazon States Language (ASL) della tua macchina a stati all'interno della console Step Functions. Per ulteriori informazioni sull'utilizzo di Workflow Studio per creare macchine a stati, consultaUtilizzo di Workflow Studio.

  > **Important**
  >
  > Ricordati di aggiornare il segnaposto Amazon Resource Name (ARN) per le risorse utilizzate nel progetto di esempio prima di eseguire il flusso di lavoro.

- Se hai selezionato Esegui una demo, Step Functions crea un progetto di esempio di sola lettura che utilizza un AWS CloudFormation modello per distribuire le AWS risorse elencate in quel modello sul tuo. Account AWS

  > **Tip**
  >
  > Per visualizzare la definizione della macchina a stati del progetto di esempio, scegli Code.

  Quando sei pronto, scegli Distribuisci ed esegui per distribuire il progetto di esempio e creare le risorse.

  La creazione di queste risorse e delle relative autorizzazioni IAM può richiedere fino a 10 minuti. Mentre le risorse vengono distribuite, puoi aprire il link CloudFormation Stack ID per vedere quali risorse vengono fornite.

  Dopo aver creato tutte le risorse del progetto di esempio, puoi vedere il nuovo progetto di esempio elencato nella pagina State machines.

> **Important**
>
> Potrebbero essere applicate tariffe standard per ogni servizio utilizzato nel CloudFormation modello.

Fase 2: Eseguire la macchina a stati

1. Nella pagina Macchine a stati, scegli il tuo progetto di esempio.
2. Nella pagina del progetto di esempio, scegli Avvia esecuzione.
3. Nella finestra di dialogo Avvia esecuzione, effettuate le seguenti operazioni:

    1. (Facoltativo) Per identificare l'esecuzione, potete specificarne un nome nella casella Nome. Per impostazione predefinita, Step Functions genera automaticamente un nome di esecuzione univoco.

        > **Note**
        >
        > Step Functions consente di creare nomi per macchine a stati, esecuzioni, attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

    2. (Facoltativo) Nella casella Input, inserisci i valori di input in formato JSON per eseguire il flusso di lavoro.

        Se avete scelto di eseguire una demo, non è necessario fornire alcun input di esecuzione.

    3. Selezionare Start execution (Avvia esecuzione).
    4. La console Step Functions ti indirizza a una pagina intitolata con il tuo ID di esecuzione. Questa pagina è nota come pagina dei dettagli di esecuzione. In questa pagina è possibile esaminare i risultati dell'esecuzione man mano che l'esecuzione procede o dopo il suo completamento.

        Per esaminare i risultati dell'esecuzione, scegliete i singoli stati nella vista Grafico, quindi scegliete le singole schede Dettagli del passaggio nel riquadro per visualizzare i dettagli di ogni stato, inclusi rispettivamente input, output e definizione. Per i dettagli sulle informazioni

sull'esecuzione che è possibile visualizzare nella pagina Dettagli di esecuzione, vedere[Pagina dei dettagli di esecuzione: panoramica dell'interfaccia](#).

Quote

AWS Step Functions impone quote alle dimensioni di determinati parametri della macchina a stati, come il numero di azioni API durante un determinato periodo di tempo o il numero di macchine a stati che è possibile definire. Sebbene queste quote siano state previste per impedire alle macchine con uno stato configurato in modo errato di utilizzare tutte le risorse di sistema, non sono rigide.

Per richiedere un aumento della quota di servizio, puoi effettuare una delle seguenti operazioni:

- Utilizzare la console Service Quotas all'indirizzo https://console.aws.amazon.com/servicequotas/home. Per informazioni sulla richiesta di un aumento della quota utilizzando la console Service Quotas, vedere Richiesta di un aumento della quota nella Service Quotas User Guide.
- Utilizza la pagina Support Center in AWS Management Console per richiedere un aumento della quota per le risorse fornite da AWS Step Functions su base regionale. Per ulteriori informazioni, consulta Service Quotas di AWS nella Riferimenti generali di AWS.

> **Note**
>
> Se una particolare fase dell'esecuzione di una macchina a stati o un'attività dura troppo, puoi configurare un timeout di macchina per attivare un evento di timeout.

Argomenti

- Quote generali
- Quote relative agli account
- Quote relative a HTTP Task
- Quote relative alla limitazione statale
- Quote relative alla limitazione delle azioni delle API
- Quote relative alle esecuzioni di macchine a stati
- Quote relative all'esecuzione delle attività
- Quote relative a versioni e alias
- Restrizioni relative all'etichettatura

Quote generali

Quota	Descrizione
Nomi in Step Functions	I nomi delle macchine a stati, delle esecuzioni e delle attività non devono superare gli 80 caratteri di lunghezza. Questi nomi devono essere univoci per l'account e la AWS regione e non devono contenere nessuno dei seguenti elementi: • Spazi bianchi • Caratteri jolly () `? *` • Caratteri tra parentesi () `< > { } [ ]` • Caratteri speciali () `" # % \ ^ | ~ ` $ & , ; : /` • Caratteri di controllo (`\\u0000`- `\\u001f` o `\\u007f` -`\\u009f`). Se la macchina a stati è di tipo Express, è possibile fornire lo stesso nome a più esecuzioni della macchina a stati. Step Functions genera un ARN di esecuzione univoco per ogni esecuzione della macchina a stati Express, anche se più esecuzioni hanno lo stesso nome. Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.

Quote relative agli account

Risorsa	Quota predefinita	Può essere aumentata fino a
Numero massimo di macchine a stati registrate	10.000	25.000
Numero massimo di attività registrate	10.000	15.000
Massima dimensione della richiesta	1 MB per richiesta. Questa è la dimensione totale dei dati per richiesta API Step Functions, inclusa l'intestazione della richiesta e tutti gli altri dati di richiesta associati.	Quota rigida
Numero massimo di esecuzioni aperte per account	1.000.000 di esecuzioni per ognuna Account AWS . Regione AWS Il superamento di questo limite causerà un errore `ExecutionLimitExceeded` . Questo non si applica a Express Workflows.	Milioni
Numero massimo di mappe aperte Una Map Run aperta è una Map Run iniziata ma non ancora completata. Le Map Run pianificate attendono l'MapRunStartedevento finché il numero totale di Map Run aperte non sarà inferiore alla quota predefinita di 1000.	1000 Questa quota è applicabile allo stato della mappa distribuita.	Quota rigida

Risorsa	Quota predefinita	Può essere aumentata fino a
Numero massimo redrivesdi mappe.	1000 Questa quota è applicabile allo stato della mappa distribuita.	Quota rigida

Quote relative a HTTP Task

Le attività HTTP vengono limitate utilizzando uno schema di token bucket per mantenere la larghezza di banda del servizio. Step Functions La tabella seguente elenca la dimensione del bucket e la frequenza di ricarica per le attività HTTP.

Risorsa	Dimensione bucket	Tasso di riempimento al secondo
Attività HTTP	300	300

La tabella seguente elenca la quota per la durata di un task HTTP.

Risorsa	Quota predefinita
Durata dell'attività HTTP La durata di un'attività HTTP si riferisce al tempo impiegato da un'attività HTTP per inviare una richiesta HTTP e ricevere una risposta.	60 secondi Si tratta di una quota fissa che non può essere modificata.

Quote relative alla limitazione statale

Le transizioni di stato di Step Functions vengono limitate utilizzando uno schema token bucket per mantenere la larghezza di banda del servizio. I flussi di lavoro standard e i flussi di lavoro Express prevedono una limitazione delle transizioni di stato diversa. Le quote Standard Workflows sono quote flessibili e possono essere aumentate.

> **Note**
>
> La limitazione della metrica del `StateTransition` servizio viene riportata come in `ExecutionThrottled` Amazon. CloudWatch Per ulteriori informazioni, consulta la metrica. `ExecutionThrottled` CloudWatch

	Standard		Express	
Metrica del servizio	Dimensione bucket	Tasso di riempimento al secondo	Dimensione bucket	Tasso di riempimento al secondo
`StateTransition` — Negli Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (Oregon) ed Europa (Irlanda)	5.000	5.000	Illimitato	Illimitato
`StateTransition` — Tutte le altre regioni	800	800	Illimitato	Illimitato

Quote relative alla limitazione delle azioni delle API

Alcune azioni dell'API Step Functions vengono limitate utilizzando uno schema token bucket per mantenere la larghezza di banda del servizio. Queste quote sono quote flessibili e possono essere aumentate.

> **Note**
>
> Le quote di limitazione si intendono per account e per regione. AWS AWS Step Functions può aumentare sia la dimensione del secchio che la velocità di ricarica in qualsiasi momento.

	Standard		Express	
Nome API	Dimensione bucket	Tasso di riempimento al secondo	Dimensione bucket	Tasso di riempimento al secondo
`StartExecution` — Negli Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (Oregon) ed Europa (Irlanda)	1.300	300	6.000	6.000
`StartExecution` — Tutte le altre regioni	800	150	6.000	6.000

Quota relativa all' TestState API

La tabella seguente elenca la quota disponibile per l' TestState API.

Nome API	Quota	Può essere aumentata fino a
TestState	1 transazione al secondo (TPS)	Quota rigida

Altre quote

Queste quote sono quote agevolate e possono essere aumentate.

	In US East (N. Virginia), US West (Oregon), and Europe (Ireland)		All other regions	
Nome API	Dimensione bucket	Tasso di riempimento al secondo	Dimensione bucket	Tasso di riempimento al secondo
`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

	In US East (N. Virginia), US West (Oregon), and Europe (Ireland)		All other regions	
Nome API	Dimensione bucket	Tasso di riempimento al secondo	Dimensione bucket	Tasso di riempimento al secondo
`GetActivityTask`	3.000	500	1.500	300
`GetExecutionHistory`	400	20	400	20
`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	
Nome API	Dimensione bucket	Tasso di riempimento al secondo	Dimensione bucket	Tasso di riempimento al secondo
StartSync Execution	Le chiamate all'API Synchrono us Express Execution non contribui scono ai limiti di capacità degli account esistenti . Step Functions fornisce capacità su richiesta e si ridimensi ona automatic amente con un carico di lavoro sostenuto. I picchi di carico di lavoro possono essere limitati fino a quando la capacità non sarà disponibile. Se si verifica una limitazio ne, riprova dopo qualche tempo. Per informazioni sui			

	In US East (N. Virginia), US West (Oregon), and Europe (Ireland)		All other regions	
Nome API	Dimensione bucket	Tasso di riempimento al secondo	Dimensione bucket	Tasso di riempimento al secondo
	flussi di lavoro Synchrono us Express, consulta. Flussi di lavoro Express sincroni e asincroni			
`StopExecution`	1.000	200	500	25
`TagResource`	200	1	200	1
`UntagResource`	200	1	200	1
`UpdateStateMachine`	100	1	100	1

Quote relative alle esecuzioni di macchine a stati

La tabella seguente descrive le quote relative alle esecuzioni di macchine a stati. Le quote di esecuzione delle macchine a stati sono quote fisse che non possono essere modificate, ad eccezione della quota relativa al tempo di conservazione della cronologia di esecuzione.

Quota	Standard	Express
Tempo massimo di esecuzione	1 anno. Se un'esecuzione viene eseguita per più del massimo di 1 anno, fallirà con un `States.Timeout` errore	5 minuti. Se un'esecuzione viene eseguita per più del massimo di 5 minuti, avrà esito negativo con un

Quota	Standard	Express
	ed emetterà una metrica. `ExecutionsTimedOut` CloudWatch	`States.Timeout` errore ed emetterà una metrica. `ExecutionsTimedOut` CloudWatch
Dimensione massima della cronologia di esecuzione	25.000 eventi in una cronologia di esecuzion e di una macchina a stato singolo. Eventuali esecuzioni successive al raggiungimento di questa quota non riusciran no. Per evitare una simile evenienza, consultare Evita di raggiungere la quota di cronologia.	Illimitate.
Tempo massimo di inattività dell'esecuzione	1 anno (vincolato dal tempo massimo di esecuzione).	5 minuti (vincolato dal tempo massimo di esecuzione).

Quota	Standard	Express
Tempo di conservazione della cronologia di esecuzione	90 giorni dopo la chiusura di un'esecuzione. Trascorso questo lasso di tempo, non sarà più possibile recuperare o visualizzare la cronologia dell'esecuzione. Non esiste un'ulteriore quota per il numero di esecuzioni chiuse che Step Functions mantiene. Per soddisfare i requisiti di conformità, organizzativi o normativi, è possibile ridurre il periodo di conservazione della cronologia di esecuzione a 30 giorni inviando una richiesta di quota. A tale scopo, utilizza AWS Support Center Console e crea un nuovo caso. La modifica per ridurre il periodo di conservazione a 30 giorni è applicabile a ogni account in una regione.	Per visualizzare la cronologia di esecuzione, è necessario configurare la registrazione di Amazon CloudWatch Logs. Per ulteriori informazioni, consulta Registrazione utilizzandoCloudWatchRegistri.

Quota	Standard	Express
Periodo di esecuzione redrivable Redrivableil periodo si riferisce al periodo durante il quale è possibile eseguire redriveun determinato flusso di lavoro standard. Questo periodo inizia dal giorno in cui una macchina a stati completa la sua esecuzione.	14 giorni. Questa quota fissa è applicabile allo stato della Mappa Distribuita.	Redriveattualmente non è supportato per i flussi di lavoro Express.

Quote relative all'esecuzione delle attività

La tabella seguente descrive le quote relative all'esecuzione delle attività e sono quote rigide che non possono essere modificate.

Quota	Standard	Express
Tempo massimo di esecuzione dell'attività	1 anno (limitato dal tempo massimo di esecuzione)	5 minuti (limitato dal tempo massimo di esecuzione)
Tempo massimo per cui Step Functions mantiene un'attività in coda	1 anno (limitato dal tempo massimo di esecuzione)	5 minuti (limitato dal tempo massimo di esecuzione)
Numero massimo di poller di attività per Amazon Resource Name (ARN)	1.000 poller che chiamano `GetActivityTask` per ARN. Il superamento di questa quota genera l'errore: "The maximum number of workers concurrently polling for activity tasks has been reached" ("È stato raggiunto il numero massimo di lavorator	Non si applica ai flussi di lavoro Express.

Quota	Standard	Express
	i contemporaneamente impegnati nel polling per task di attività").	
Dimensione massima di input o output per un'attività, uno stato o un'esecuzione	256 KB di dati come stringa con codifica UTF-8. Questa quota influisce sulle attività (attività, funzione Lambda o servizio integrato), sullo stato o sull'output di esecuzione e sui dati di input durante la pianificazione di un'attività, l'immissione di uno stato o l'avvio di un'esecuzione.	256 KB di dati come stringa con codifica UTF-8. Questa quota influisce sulle attività (attività, funzione Lambda o servizio integrato), sullo stato o sull'output di esecuzione e sui dati di input durante la pianificazione di un'attività, l'immissione di uno stato o l'avvio di un'esecuzione.

Quote relative a versioni e alias

Risorsa	Quota predefinita
Numero massimo di versioni pubblicate di macchine a stati	1000 per ogni macchina a stati. Per richiedere un aumento di questo limite flessibile, utilizza la pagina Support Center nel AWS Management Console.
Numero massimo di alias di macchine a stati	100 per ogni macchina a stati. Per richiedere un aumento di questo limite flessibile, utilizza la pagina Support Center nel AWS Management Console.

Restrizioni relative all'etichettatura

Tieni presente queste restrizioni quando tagghi le risorse Step Functions.

> **Note**
>
> Le quote di tagging non possono essere aumentate come le altre.

Limitazione	Descrizione
Numero massimo di tag per risorsa	50
Lunghezza massima chiave	128 caratteri Unicode in UTF-8
Lunghezza massima del valore	256 caratteri Unicode in UTF-8
Limitazione dei prefissi	Non utilizzate il `aws:` prefisso nei nomi o nei valori dei tag perché è AWS riservato all'uso. Non è possibile modificare né eliminare i nomi o i valori di tag con tale prefisso. I tag con questo prefisso non vengono conteggiati per la quota di tag per risorsa.
Limitazioni di caratteri	I tag possono contenere solo lettere unicode, cifre, spazi vuoti o questi simboli: `_ . : / = + - @`

Registrazione e monitoraggio in AWS Step Functions

La registrazione e il monitoraggio sono importanti per mantenere l'affidabilità, la disponibilità e le prestazioni di Step Functions e delle vostre AWS soluzioni. Sono disponibili diversi strumenti da utilizzare con Step Functions:

> **Tip**
>
> Per implementare un flusso di lavoro di esempio Account AWS e imparare a monitorare metriche, registri e tracce dell'esecuzione del flusso di lavoro, consulta il Modulo 12 - Osservabilità del Workshop. AWS Step Functions

Argomenti

- Monitoraggio di Step Functions tramite CloudWatch
- EventBridge (CloudWatch Eventi) per le modifiche allo stato di esecuzione di Step Functions
- Funzioni della fase di registrazione tramite AWS CloudTrail
- Registrazione utilizzandoCloudWatchRegistri
- AWS X-Ray e Step Functions
- Uso di Notifiche all'utente AWS con AWS Step Functions

Monitoraggio di Step Functions tramite CloudWatch

Il monitoraggio è una parte importante per mantenere l'affidabilità, la disponibilità e le prestazioni delle AWS Step Functions tue AWS soluzioni. È necessario raccogliere il maggior numero di dati di monitoraggio dai AWS servizi utilizzati in modo da poter eseguire il debug di errori multipunto. Prima di iniziare a monitorare Step Functions, è necessario creare un piano di monitoraggio che risponda alle seguenti domande:

- Quali sono gli obiettivi del monitoraggio?
- Di quali risorse si intende eseguire il monitoraggio?
- Con quale frequenza sarà eseguito il monitoraggio di queste risorse?
- Quali strumenti di monitoraggio verranno utilizzati?
- Chi eseguirà i processi di monitoraggio?

- Chi deve ricevere una notifica quando si verifica un problema?

Il passaggio successivo consiste nello stabilire una baseline per le prestazioni normali di nel tuo ambiente. A questo proposito, misura le prestazioni in diversi orari e in diverse condizioni di carico. Durante il monitoraggio di Step Functions, valuta la possibilità di archiviare i dati di monitoraggio storici. Tali dati possono servire come baseline per il confronto con i dati relativi sulle prestazioni correnti, per identificare modelli di prestazioni normali e anomalie nelle prestazioni nonché elaborare metodi mediante i quali risolvere i problemi.

Ad esempio, con Step Functions, puoi monitorare quante attività o AWS Lambda attività falliscono a causa di un timeout del battito cardiaco. Quando le prestazioni non rientrano nella baseline stabilita, è possibile che sia necessario modificare l'intervallo di heartbeat.

Per stabilire una baseline, devi monitorare almeno i seguenti parametri:

- `ActivitiesStarted`
- `ActivitiesTimedOut`
- `ExecutionsStarted`
- `ExecutionsTimedOut`
- `LambdaFunctionsStarted`
- `LambdaFunctionsTimedOut`

Le seguenti sezioni descrivono i parametri che Step Functions fornisce ad Amazon CloudWatch. Puoi utilizzare questi parametri per monitorare le macchine a stati e le attività e per impostare allarmi sui valori di soglia. Puoi visualizzare le metriche utilizzando. AWS Management Console

Metriche che segnalano un intervallo di tempo

Alcune delle CloudWatch metriche di Step Functions sono intervalli di tempo, misurati sempre in millisecondi. Queste metriche generalmente corrispondono alle fasi dell'esecuzione per le quali è possibile impostare i timeout della macchina a stati, dell'attività e della funzione Lambda, con nomi descrittivi.

Ad esempio, il parametro `ActivityRunTime` misura il tempo necessario per il completamento di un'attività dopo che ne viene avviata l'esecuzione. Puoi impostare un valore di timeout per lo stesso periodo di tempo.

Nella CloudWatch console, puoi ottenere i risultati migliori se scegli la media come statistica di visualizzazione per le metriche degli intervalli di tempo.

Metriche che riportano un conteggio

Alcune CloudWatch metriche di Step Functions riportano i risultati sotto forma di conteggio. Ad esempio, `ExecutionsFailed` registra il numero di esecuzioni di macchina di stato che hanno esito negativo.

Step Functions emette due `ExecutionsStarted` metriche per ogni esecuzione di una macchina a stati. Ciò fa sì che la SampleCountstatistica della `ExecutionsStarted` metrica mostri il valore di 2 per ogni esecuzione di una macchina a stati. La SampleCount statistica mostra `ExecutionStarted=1` e `ExecutionStarted=0` quando l'esecuzione è completata.

> **Tip**
>
> Ti consigliamo di selezionare Sum come statistica di visualizzazione per le metriche che riportano un conteggio nella console. CloudWatch

Parametri di esecuzione

Il `AWS/States` namespace include le seguenti metriche per tutte le esecuzioni di Step Functions. Si tratta di metriche adimensionali che si applicano a tutto il tuo account in una regione.

Parametro	Descrizione
`OpenExecutionCount`	Numero approssimativo di esecuzioni attualmente aperte, ossia flussi di lavoro attualmente in corso nel tuo account. L'intento è quello di fornire informazioni dettagliate su quando i flussi di lavoro si avvicinano al limite massimo di esecuzione, per evitare ExecutionLimitExceedederrori durante le chiamate o per i flussi di lavoro standard. `StartExecution` `RedriveExecution` La metrica dipende dalle transizioni di stato del flusso di lavoro attive, pertanto, a livelli bassi, la stima potrebbe non essere in linea con il numero di flussi di lavoro in esecuzione osservato.

Parametro	Descrizione
`OpenExecutionLimit`	Numero massimo di esecuzioni aperte. Per ulteriori informazioni, consulta Quote relative agli account. Questo limite non si applica ai flussi di lavoro Express.

Metriche di esecuzione per la macchina a stati con versione o alias

Quando si esegue l'esecuzione di una macchina a stati con una versione o un alias, Step Functions emette le seguenti metriche. La `ExecutionThrottled` metrica verrà emessa solo in caso di esecuzione limitata. Queste metriche includeranno un metodo `StateMachineArn` per identificare una macchina a stati specifica.

Parametro	Descrizione
`ExecutionTime`	Intervallo, in millisecondi, tra l'inizio dell'esecuzione e il momento in cui si chiude.
`ExecutionThrottled`	Numero di `StateEntered` eventi e tentativi che sono stati limitati. Questo si riferisce al throttling di `StateTransition` . Per ulteriori informazioni, consulta Quote relative alla limitazione statale.
`ExecutionsAborted`	Numero di esecuzioni interrotte o terminate.
`ExecutionsFailed`	Numero di esecuzioni non riuscite.
`ExecutionsStarted`	Numero di esecuzioni iniziate.
`ExecutionsSucceeded`	Numero di esecuzioni completate con successo.
`ExecutionsTimedOut`	Numero di esecuzioni scadute per qualsiasi motivo.

Metriche di esecuzione per Express Workflows

Il `AWS/States` namespace include le seguenti metriche per le esecuzioni di Step Functions Express Workflows.

Parametro	Descrizione
`ExpressExecutionMemory`	La memoria totale consumata da un Express Workflow.
`ExpressExecutionBilledDuration`	La durata per la quale viene addebitato un Express Workflow.
`ExpressExecutionBilledMemory`	La quantità di memoria consumata per la quale viene addebitato un Express Workflow.

Redrivemetriche di esecuzione per i flussi di lavoro standard

Quando si esegue redriveuna macchina a stati, Step Functions emette le seguenti metriche.

Per tutte le redriven esecuzioni, viene emessa la `Executions*` metrica. Ad esempio, supponiamo che un'esecuzione si interromparedriven. Questa esecuzione emetterà punti dati diversi da zero per entrambi e. `RedrivenExecutionsAborted` `ExecutionsAborted`

Parametro	Descrizione
`ExecutionsRedriven`	redrivenNumero di esecuzioni.
`RedrivenExecutionsAborted`	Numero di redriven esecuzioni annullate o terminate.
`RedrivenExecutionsTimedOut`	Numero di redriven esecuzioni scadute per qualsiasi motivo.
`RedrivenExecutionsSucceeded`	Numero di redriven esecuzioni completate con successo.
`RedrivenExecutionsFailed`	Numero di redriven esecuzioni non riuscite.

Metriche di esecuzione di Dimension for Step Functions

Dimensione	Descrizione
`StateMachineArn`	L'Amazon Resource Name (ARN) della macchina a stati per l'esecuzione in questione.

Dimensioni per le esecuzioni con versione

Dimensione	Descrizione
`StateMachineArn`	L'Amazon Resource Name (ARN) della macchina a stati la cui esecuzione è stata avviata da una versione.
`Version`	Versione della macchina a stati utilizzata per avviare l'esecuzione.

Dimensioni per le esecuzioni con un alias

Dimensione	Descrizione
`StateMachineArn`	L'Amazon Resource Name (ARN) della macchina a stati la cui esecuzione è stata avviata da un alias.
`Alias`	Alias della macchina a stati utilizzato per avviare l'esecuzione.

Metriche del conteggio delle risorse per versioni e alias

Il `AWS/States` namespace include le seguenti metriche per il conteggio delle versioni e degli alias di una macchina a stati.

Parametro	Descrizione
AliasCount	Numero di alias creati per la macchina a stati.

Parametro	Descrizione
	È possibile creare fino a 100 alias per ogni macchina a stati.
VersionCount	Numero di versioni pubblicate per la macchina a stati. È possibile pubblicare fino a 1000 versioni di una macchina a stati.

Dimensione per le metriche del conteggio delle risorse per versioni e alias

Dimensione	Descrizione
`ResourceArn`	L'Amazon Resource Name (ARN) della macchina a stati con una versione o un alias.

Parametri delle attività

Il `AWS/States` namespace include le seguenti metriche per le attività di Step Functions.

Parametro	Descrizione
`ActivityRunTime`	Intervallo, in millisecondi, tra l'inizio dell'attività e il momento in cui si chiude.
`ActivityScheduleTime`	Intervallo, in millisecondi, per il quale l'attività rimane nello stato di pianificazione.
`ActivityTime`	Intervallo, in millisecondi, tra il momento in cui l'attività è pianifica ta e l'ora di chiusura.
`ActivitiesFailed`	Numero di attività non riuscite.
`ActivitiesHeartbea tTimedOut`	Numero di attività che scadono a causa di un timeout del battito cardiaco.
`ActivitiesScheduled`	Numero di attività programmate.

Parametro	Descrizione
`ActivitiesStarted`	Numero di attività avviate.
`ActivitiesSucceeded`	Numero di attività completate con successo.
`ActivitiesTimedOut`	Numero di attività che scadono alla chiusura.

Metriche delle attività di Dimension for Step Functions

Dimensione	Descrizione
`ActivityArn`	L'ARN dell'attività.

Parametri della funzione Lambda

Il `AWS/States` namespace include le seguenti metriche per le funzioni Step Functions Lambda.

Parametro	Descrizione
`LambdaFunctionRunTime`	Intervallo, in millisecondi, tra l'inizio e la chiusura della funzione Lambda.
`LambdaFunctionScheduleTime`	Intervallo, in millisecondi, per il quale la funzione Lambda rimane nello stato di pianificazione.
`LambdaFunctionTime`	Intervallo, in millisecondi, tra il momento in cui la funzione Lambda è pianificata e l'ora di chiusura.
`LambdaFunctionsFailed`	Numero di funzioni Lambda non riuscite.
`LambdaFunctionsScheduled`	Numero di funzioni Lambda pianificate.
`LambdaFunctionsStarted`	Numero di funzioni Lambda avviate.

Parametro	Descrizione
`LambdaFunctionsSucceeded`	Numero di funzioni Lambda completate con successo.
`LambdaFunctionsTimedOut`	Numero di funzioni Lambda che scadono alla chiusura.

Metriche della funzione Lambda di Dimension for Step Functions

Dimensione	Descrizione
`LambdaFunctionArn`	ARN della funzione Lambda.

> **Note**
>
> Le metriche delle funzioni Lambda vengono emesse per gli stati delle attività che specificano l'ARN della funzione Lambda nel campo. `Resource` Stati di attività che utilizzano invece `"Resource": "arn:aws:states:::lambda:invoke"` emit Service Integration Metrics. Per ulteriori informazioni, consulta Richiama Lambda con Step Functions.

Parametri di integrazione del servizio

Il `AWS/States` namespace include le seguenti metriche per le integrazioni dei servizi Step Functions. Per ulteriori informazioni, consulta Utilizzo AWS Step Functions con altri servizi.

Parametro	Descrizione
`ServiceIntegrationRunTime`	Intervallo, in millisecondi, tra l'inizio del Service Task e il momento in cui viene chiuso.
`ServiceIntegrationScheduleTime`	Intervallo, in millisecondi, durante il quale il Service Task rimane nello stato di pianificazione.

Parametro	Descrizione
`ServiceIntegrationTime`	Intervallo, in millisecondi, tra il momento in cui il Service Task è pianificato e l'ora di chiusura.
`ServiceIntegrationsFailed`	Numero di attività di servizio non riuscite.
`ServiceIntegrationsScheduled`	Numero di attività di assistenza pianificate.
`ServiceIntegrationsStarted`	Numero di attività di servizio avviate.
`ServiceIntegrationsSucceeded`	Numero di attività di assistenza completate con successo.
`ServiceIntegrationsTimedOut`	Numero di attività di servizio scadute alla chiusura.

Metriche di integrazione dei servizi Dimension for Step Functions

Dimensione	Descrizione
`ServiceIntegrationResourceArn`	L’ARN della risorsa del servizio integrato.

Parametri servizio

Il `AWS/States` namespace include le seguenti metriche per il servizio Step Functions.

Parametro	Descrizione
`ThrottledEvents`	Numero di richieste che sono state limitate.
`ProvisionedBucketSize`	Numero di richieste disponibili al secondo.

Parametro	Descrizione
`ProvisionedRefillRate`	Numero di richieste al secondo consentite nel bucket.
`ConsumedCapacity`	Numero di richieste al secondo.

Metriche del servizio Dimension for Step Functions

Dimensione	Descrizione
`ServiceMetric`	Filtra i dati per visualizzare i parametri delle transizioni dello stato.

Parametri API

Il `AWS/States` namespace include le seguenti metriche per l'API Step Functions.

Parametro	Descrizione
`ThrottledEvents`	Numero di richieste che sono state limitate.
`ProvisionedBucketSize`	Numero di richieste disponibili al secondo.
`ProvisionedRefillRate`	Numero di richieste al secondo consentite nel bucket.
`ConsumedCapacity`	Numero di richieste al secondo.

Metriche dell'API Dimension for Step Functions

Dimensione	Descrizione
`APIName`	Filtra i dati in base a un'API con il nome API specificato.

Fornitura delle metriche Best-Effort CloudWatch

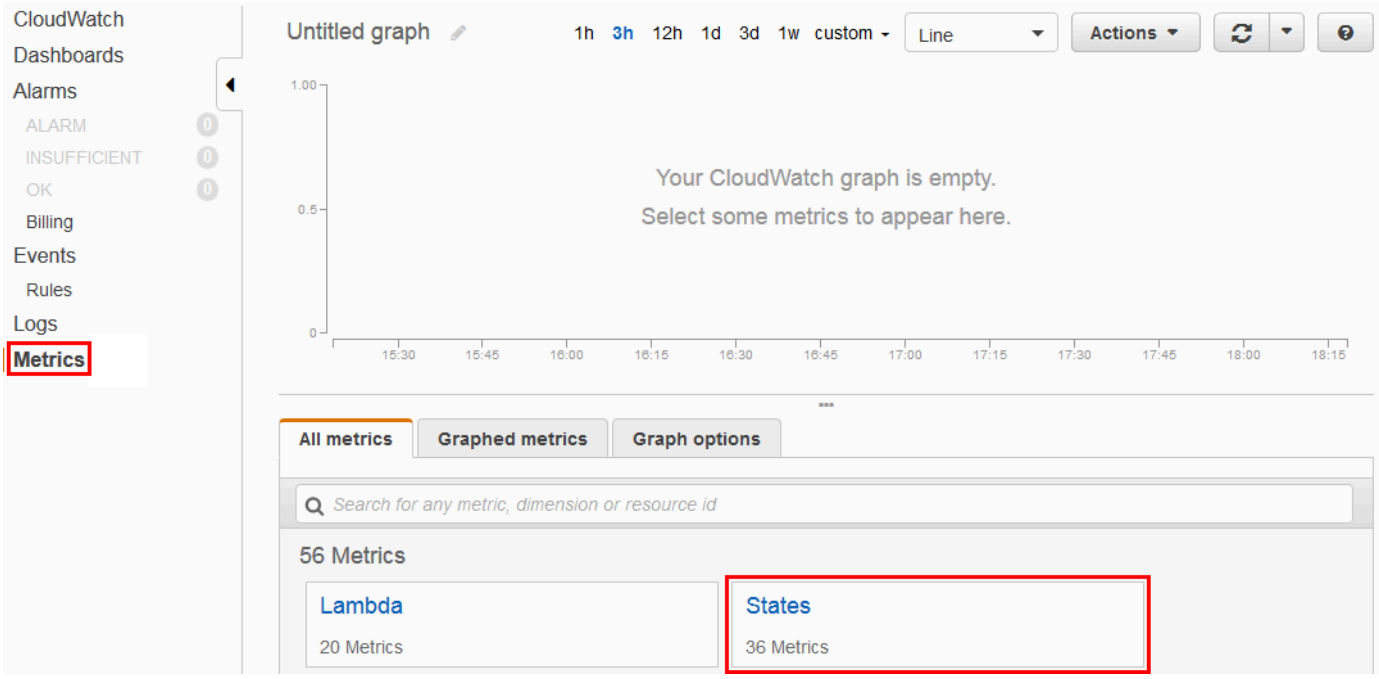
I parametri di CloudWatch vengono distribuiti sulla base del miglior tentativo.

La completezza e la tempestività dei parametri non è garantita. È possibile che il data point per una richiesta specifica venga restituito con un timestamp successivo a quello del momento effettivo di elaborazione della richiesta. Il data point potrebbe subire un ritardo di un minuto prima di essere disponibile oppure potrebbe non essere consegnato affatto. CloudWatch CloudWatchle metriche di richiesta ti danno un'idea delle esecuzioni delle macchine a stati in tempo quasi reale. Non è pensato per essere una contabilità completa di tutte le metriche relative all'esecuzione.

Poiché questa funzionalità richiede il massimo impegno, i report disponibili nella dashboard di Billing & Cost Management potrebbero includere una o più richieste di accesso che non compaiono nelle metriche di esecuzione.

Visualizzazione delle metriche per Step Functions

1. Accedi a AWS Management Console e apri la CloudWatch console.
2. Scegliere Metrics (Parametri) e nella scheda All Metrics (Tutti i parametri), scegliere States (Stati).

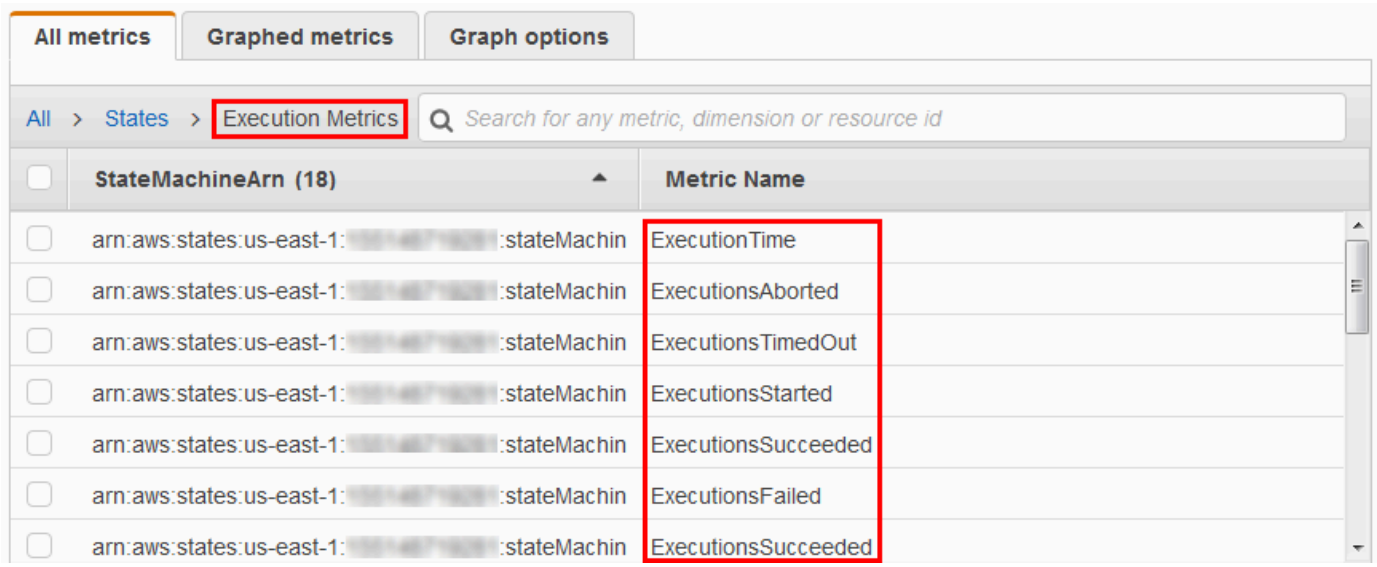


Se hai eseguito esecuzioni di recente, vedrai fino a quattro tipi di metriche:

- Parametri di esecuzione

- Parametri di funzioni di attività
- Metriche della funzione Lambda
- Metriche di integrazione dei servizi

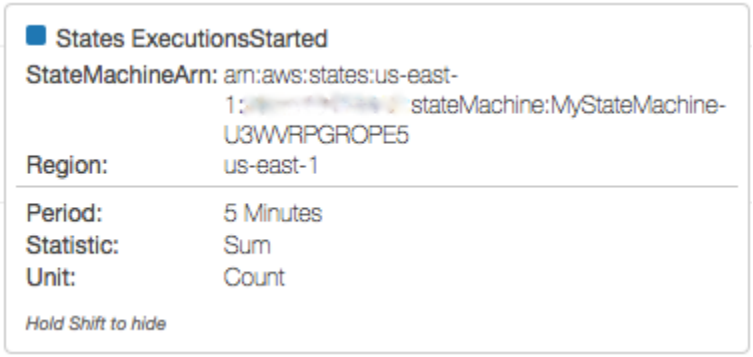
3. Scegliere un tipo di parametro per visualizzare un elenco di parametri.



- Per ordinare le metriche in base al nome della metrica oppure StateMachineArn, utilizza le intestazioni delle colonne.
- Per visualizzare grafici per un parametro, selezionare la casella accanto al parametro nell'elenco. È possibile modificare i parametri del grafico utilizzando i controlli degli intervalli di tempo sopra il grafico.

  È possibile scegliere intervalli di tempo personalizzati utilizzando valori assoluti o relativi (giorni e ore specifici). È anche possibile utilizzare l'elenco a discesa per visualizzare i valori come linee, aree impilate o numeri (valori).
- Per visualizzare i dettagli relativi a un grafico, passare il mouse sul codice colore del parametro visualizzato sotto il grafico.

  Vengono visualizzati i dettagli del parametro.



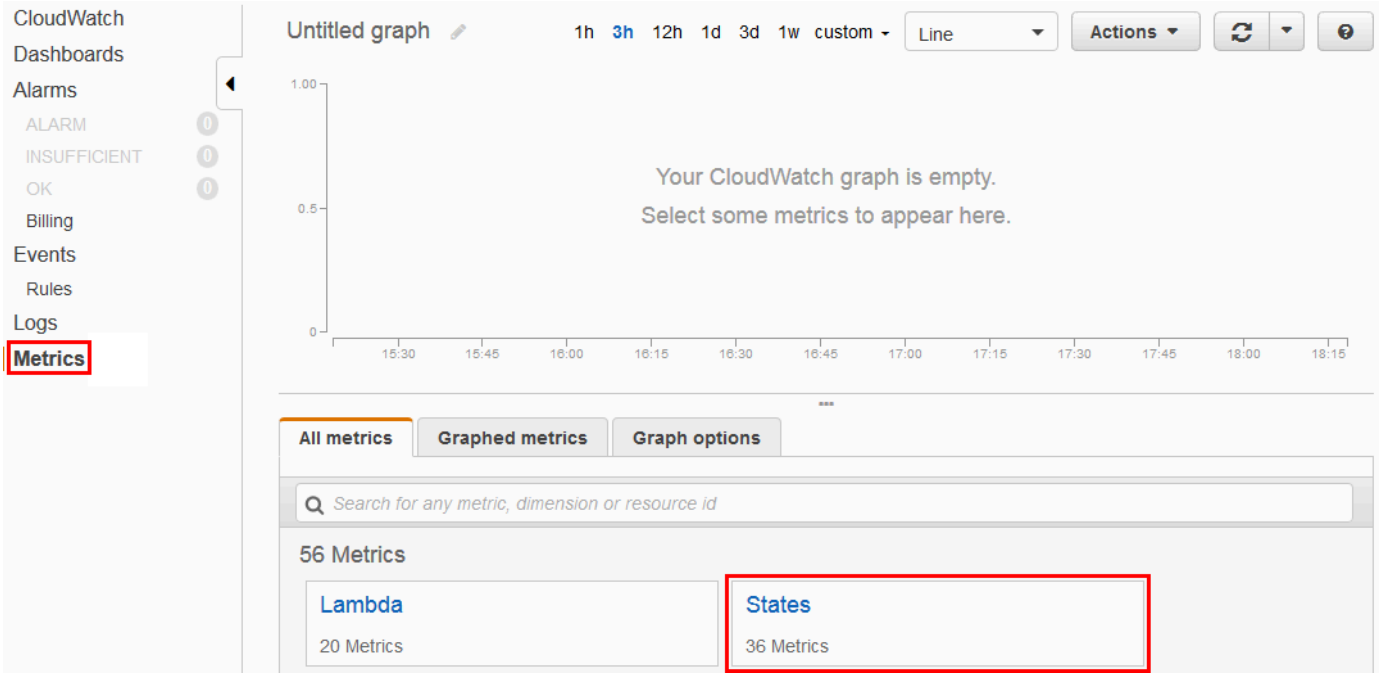
Per ulteriori informazioni sull'utilizzo dei CloudWatch parametri, consulta Using Amazon CloudWatch Metrics nella Amazon CloudWatch User Guide.

Impostazione degli allarmi per Step Functions

Puoi utilizzare gli CloudWatch allarmi Amazon per eseguire azioni. Ad esempio, se desideri sapere quando viene raggiunta una soglia di allarme, puoi impostare un allarme per inviare una notifica a un argomento di Amazon SNS o inviare un'e-mail quando la `StateMachinesFailed` metrica supera una determinata soglia.

Per impostare un allarme per un parametro

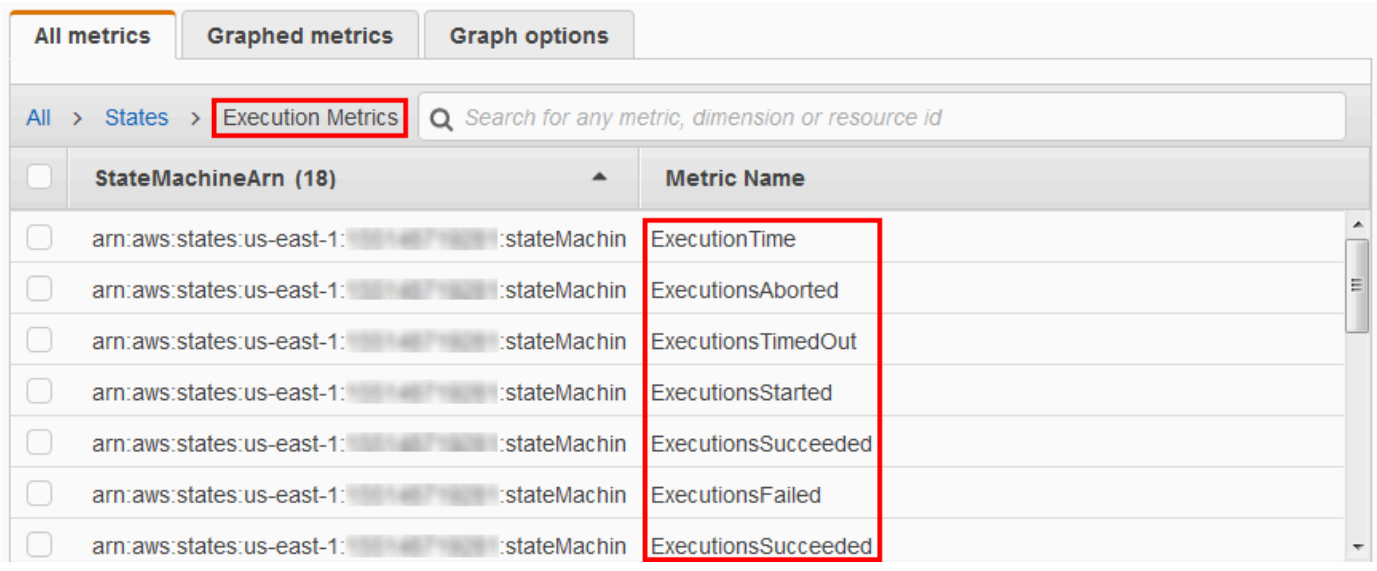
1. Accedi AWS Management Console e apri la CloudWatch console.
2. Scegliere Metrics (Parametri) e nella scheda All Metrics (Tutti i parametri), scegliere States (Stati).



Se hai eseguito esecuzioni di recente, vedrai fino a quattro tipi di metriche:

- Parametri di esecuzione
- Parametri di funzioni di attività
- Metriche della funzione Lambda
- Metriche di integrazione dei servizi

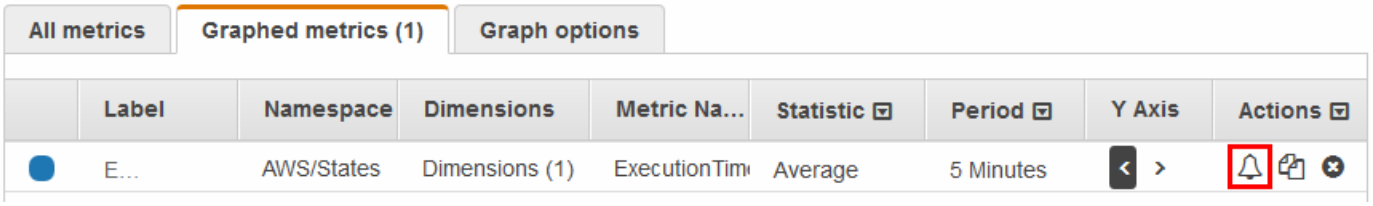
3. Scegliere un tipo di parametro per visualizzare un elenco di parametri.



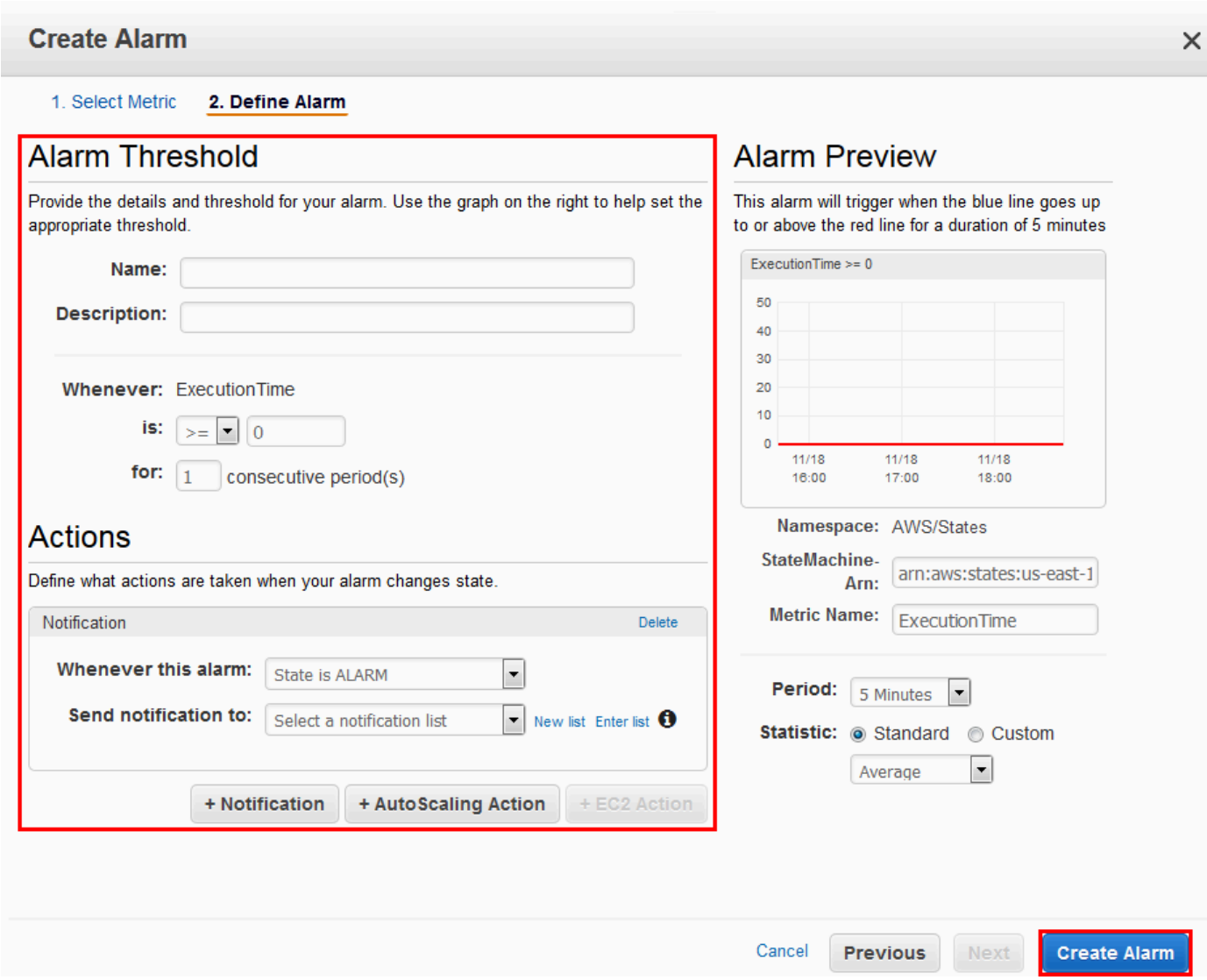
4. Scegliere un parametro, quindi Graphed metrics (Parametri nel grafico).

5. Scegliere

   accanto a un parametro nell'elenco.



   Viene visualizzata la pagina Create Alarm (Crea allarme).



6. Immettere i valori in Alarm threshold (Soglia allarme) e Actions (Operazioni), quindi scegliere Create Alarm (Crea allarme).

Per ulteriori informazioni sull'impostazione e l'utilizzo degli CloudWatch allarmi, consulta Creating Amazon CloudWatch Alarms nella Amazon CloudWatch User Guide.

EventBridge (CloudWatch Eventi) per le modifiche allo stato di esecuzione di Step Functions

Amazon EventBridge è un AWS servizio che ti consente di rispondere ai cambiamenti di stato di una AWS risorsa. Ad esempio, è possibile rispondere alle modifiche dello stato di esecuzione di un flusso di lavoro Step Functions Standard EventBridge utilizzando i due modi seguenti:

- È possibile configurare EventBridge regole per reagire agli eventi che vengono emessi quando cambia lo stato di esecuzione di una macchina a stati Step Functions. In questo modo è possibile monitorare i flussi di lavoro senza la necessità di eseguire il polling costantemente utilizzando l'API DescribeExecution. In base alle modifiche nelle esecuzioni delle macchine a stati, puoi utilizzare un EventBridge target per avviare nuove esecuzioni di macchine a stati, chiamare AWS Lambda funzioni, pubblicare messaggi su argomenti di Amazon Simple Notification Service (Amazon SNS) e altro ancora.
- È inoltre possibile configurare una macchina a stati Step Functions come destinazione in EventBridge. Ciò consente di attivare l'esecuzione di un flusso di lavoro Step Functions in risposta a un evento di un altro AWS servizio.

  Per ulteriori informazioni, consulta la Amazon EventBridge User Guide.

I flussi di lavoro Express, tuttavia, non emettono eventi a. EventBridge Per monitorare l'esecuzione di un Express Workflow, puoi utilizzare CloudWatch Logs. A tale scopo, nella pagina Dettagli di esecuzione della macchina a stati, selezionare le schede Monitoraggio e Registrazione. Nella scheda Monitoraggio, puoi visualizzare le CloudWatch metriche degli eventi, come Durata dell'esecuzione, Errori di esecuzione e Memoria fatturata. Nella scheda Registrazione, è possibile visualizzare i log recenti e la configurazione della registrazione.

> **Tip**
>
> Per distribuire un esempio di Express Workflow su Express Workflows Account AWS e imparare a monitorare Express Workflows, consulta il modulo Monitoring Express Workflows di The Workshop. AWS Step Functions

EventBridge carichi utili

Un EventBridge evento può contenere una proprietà di input nella sua definizione. Per alcuni eventi, un EventBridge evento può contenere anche una proprietà di output nella sua definizione.

- Se l'input con escape combinato e l'output con escape inviato a EventBridge supera i 248 KB, l'input verrà escluso. Analogamente, se l'output con escape supera i 248 KB, l'output verrà escluso. Questo è il risultato delle quote degli EventBridge eventi.
- È possibile determinare se un payload è stato troncato con le proprietà and. `inputDetails` `outputDetails` Per ulteriori informazioni, consulta la sezione Tipo di dati. `CloudWatchEventsExecutionDataDetails`
- Per i flussi di lavoro standard, puoi visualizzare l'input e l'output completi utilizzando DescribeExecution.
- `DescribeExecution`non è disponibile per Express Workflows. Se desideri visualizzare l'ingresso/uscita completo, puoi inserire Express Workflow in un Workflow standard. Un'altra opzione è utilizzare gli ARN Amazon S3. Per informazioni sull'uso degli ARN, consulta. the section called “Usa gli ARN Amazon S3 invece di trasferire payload di grandi dimensioni”

Argomenti

- Esempi di eventi Step Functions
- Instradamento di un evento Step Functions EventBridge nella console EventBridge

Esempi di eventi Step Functions

Di seguito sono riportati alcuni esempi di invio di eventi da parte di Step Functions a EventBridge:

Argomenti

- L'esecuzione è iniziata
- Esecuzione riuscita
- Esecuzione non riuscita
- Esecuzione scaduta
- Esecuzione interrotta

In ogni caso, la sezione `detail` in caso di dati fornisce le stesse informazioni dell'API `DescribeExecution`. Il campo `status` indica lo stato di esecuzione nel momento in cui l'evento è stato inviato, `RUNNING`, `SUCCEEDED`, `FAILED`, `TIMED_OUT` o `ABORTED`, a seconda dell'evento emesso.

L'esecuzione è iniziata

```
{
    "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
    }
}
```

Esecuzione riuscita

```
{
    "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
        }
    }
}
```

Esecuzione non riuscita

```
{
    "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
    }
}
```

Esecuzione scaduta

```
{
    "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
    }
}
```

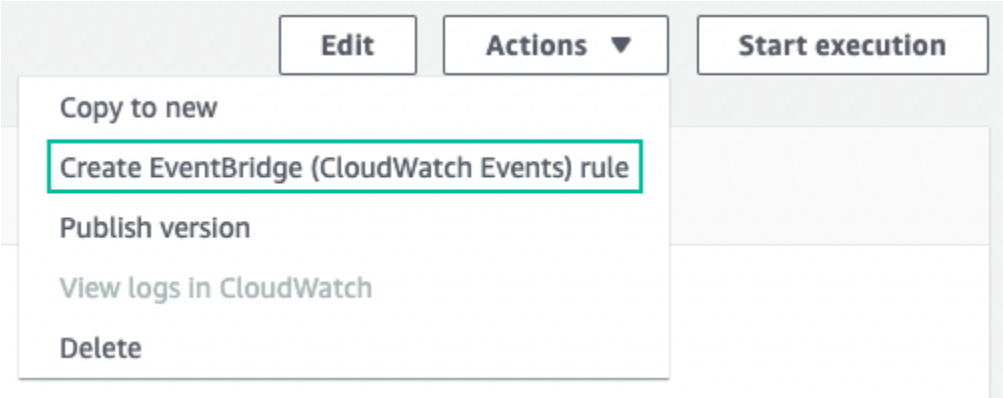
Esecuzione interrotta

```
{
    "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
    }
}
```

Instradamento di un evento Step Functions EventBridge nella console EventBridge

Utilizza le seguenti istruzioni per imparare a attivare l'esecuzione di una macchina a stati Step Functions ogni volta che una specifica macchina a stati Step Functions completa l'esecuzione con successo. Utilizza la EventBridge console Amazon per specificare la macchina a stati di cui desideri attivare l'esecuzione.

1. Nella pagina Dettagli di una macchina a stati, scegli Azioni, quindi scegli Crea regola EventBridge (CloudWatch Eventi).



   In alternativa, apri la EventBridge console all'indirizzo https://console.aws.amazon.com/events/. Nel riquadro di navigazione, scegli Regole in Autobus.
2. Scegli Crea regola. Verrà visualizzata la pagina di dettaglio Definisci regola.
3. Inserisci un nome per la regola (ad esempio,*`StepFunctionsEventRule`*) e facoltativamente inserisci una descrizione per la regola.
4. Per Event bus e Rule type, mantieni le selezioni predefinite.
5. Seleziona Avanti. Verrà aperta la pagina Build event pattern.
6. In Event Source, mantieni la selezione predefinita di AWS eventi o eventi EventBridge partner.
7. Mantieni le selezioni predefinite per le sezioni Sample event e Creation method.
8. In Event pattern, effettuate le seguenti operazioni:

   a. Nell'elenco a discesa Origine degli eventi, mantieni la selezione predefinita dei AWS servizi.

   b. Dall'elenco a discesa dei AWS servizi, scegli Step Functions.

   c. Dall'elenco a discesa Tipo di evento, selezionare Step Functions Execution Status Change.

d. (Facoltativo) Configura uno stato specifico, una macchina a stati Amazon Resource Name (ARN) o un ARN di esecuzione. Per questa procedura, scegli Stato/i specifico/i, quindi scegli SUCCESSED dall'elenco a discesa.

9. Seleziona Avanti. Verrà aperta la pagina Seleziona obiettivi.
10. In Tipi di destinazione, mantieni la selezione predefinita del AWS servizio.
11. Dall'elenco a discesa Seleziona una destinazione, scegli un AWS servizio. Ad esempio, è possibile avviare una funzione Lambda o eseguire una macchina a stati Step Functions. Per questa procedura, selezionare la macchina a stati Step Functions.
12. Dall'elenco a discesa Macchina a stati, scegli una macchina a stati.
13. In Ruolo di esecuzione, mantieni la selezione predefinita di Crea un nuovo ruolo per questa risorsa specifica.
14. Seleziona Avanti. Si apre la pagina Configura i tag.
15. scegli nuovamente Avanti. Si apre la pagina Rivedi e crea.
16. Rivedi i dettagli della regola e scegli Create rule (Crea regola).

    La regola viene creata e viene visualizzata la pagina Regole, che elenca tutte le EventBridge regole Amazon.

Funzioni della fase di registrazione tramite AWS CloudTrail

Step Functions è integrato con AWS CloudTrail un servizio che fornisce una registrazione delle azioni intraprese da un utente, da un ruolo o da un AWS servizio in Step Functions. CloudTrailacquisisce tutte le chiamate API per Step Functions come eventi, comprese le chiamate dalla console Step Functions e le chiamate di codice alle API Step Functions.

Se crei un percorso, puoi abilitare la distribuzione continua di CloudTrail eventi a un bucket Amazon Simple Storage Service (Amazon S3), inclusi gli eventi per Step Functions. Se non configuri un trail, è comunque possibile visualizzare gli eventi più recenti nella console di CloudTrail in Event history (Cronologia eventi).

Utilizzando le informazioni raccolte daCloudTrail, è possibile determinare la richiesta effettuata a Step Functions, l'indirizzo IP da cui è stata effettuata la richiesta, chi ha effettuato la richiesta, quando è stata effettuata e ulteriori dettagli.

Per ulteriori informazioni su CloudTrail, consulta la Guida per l'utente di AWS CloudTrail.

Informazioni su Step Functions in CloudTrail

CloudTrail è abilitato sull'account AWS al momento della sua creazione. Quando si verifica un'attività in Step Functions, tale attività viene registrata in un CloudTrail evento con altri eventi AWS di servizio in Cronologia eventi.

È possibile visualizzare, cercare e scaricare gli eventi recenti nell'account AWS. Per ulteriori informazioni, consulta Visualizzazione di eventi mediante la cronologia eventi di CloudTrail.

Per una registrazione continua degli eventi nel tuo AWS account, inclusi gli eventi per Step Functions, crea un percorso. Un percorso consente di CloudTrail inviare file di registro a un bucket Amazon S3. Per impostazione predefinita, quando si crea un trail nella console, il trail sarà valido in tutte le regioni AWS. Il trail registra gli eventi di tutte le Regioni nella partizione AWS e distribuisce i file di log nel bucket Amazon S3 specificato. Inoltre, puoi configurare altri servizi AWS per analizzare con maggiore dettaglio e usare i dati raccolti nei log CloudTrail.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Panoramica della creazione di un percorso
- Servizi e integrazioni CloudTrail supportati
- Configurazione delle notifiche Amazon SNS per CloudTrail
- Ricezione di file di log CloudTrail da più regioni e Ricezione di file di log CloudTrail da più account

Ogni evento o voce di log contiene informazioni sull'utente che ha generato la richiesta. Le informazioni di identità consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con le credenziali utente o root.
- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro servizio AWS.

Per ulteriori informazioni, consulta Elemento CloudTrail userIdentity.

Esempio: voci del file di registro di Step Functions

Un percorso è una configurazione che consente la distribuzione di eventi come i file di log in un bucket Amazon S3 specificato. I file di log di CloudTrail contengono una o più voci di log. Un evento rappresenta una singola richiesta da un'fonte e include informazioni sul operazione richiesta, data

e ora dell'operazione, parametri richiesti e così via. CloudTrailI file di registro non sono una traccia ordinata dello stack delle chiamate API pubbliche, quindi non vengono visualizzati in un ordine specifico.

CreateActivity

L'esempio seguente mostra una voce di log di CloudTrail che illustra l'operazione `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

L'esempio seguente mostra una voce di log di CloudTrail che illustra l'operazione `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

L'esempio seguente mostra una voce di log di CloudTrail che illustra l'operazione `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

L'esempio seguente mostra una voce di log di CloudTrail che illustra l'operazione `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

L'esempio seguente mostra una voce di log di CloudTrail che illustra l'operazione `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

L'esempio seguente mostra una voce di log di CloudTrail che illustra l'operazione `StopExecution`.

```
{
    "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"
}
```

Registrazione utilizzandoCloudWatchRegistri

I flussi di lavoro standard registrano la cronologia delle esecuzioni inAWS Step Functions, sebbene sia possibile configurare facoltativamente la registrazione su AmazonCloudWatchTronchi.

Analogamente ai flussi di lavoro standard, i flussi di lavoro Express non registrano la cronologia di esecuzione in AWS Step Functions. Per visualizzare la cronologia delle esecuzioni e i risultati di un Express Workflow, devi configurare la registrazione su AmazonCloudWatchTronchi. I log di pubblicazione non bloccano o rallentano le esecuzioni.

> **Note**
>
> Quando configuri la registrazione,CloudWatchRegistra gli addebitiverrà applicata e ti verrà addebitata la tariffa dei log venduti. Per ulteriori informazioni, vedereTronchi vendutisotto ilRegistrischeda sulCloudWatchPagina dei prezzi.

Configurazione della registrazione

Quando si crea un flusso di lavoro standard utilizzando la console Step Functions, questo non verrà configurato per abilitare la registrazione suCloudWatchTronchi. Per impostazione predefinita, un Express Workflow creato utilizzando la console Step Functions verrà configurato per abilitare la registrazione suCloudWatchTronchi.

Per i flussi di lavoro Express, Step Functions può creare un ruolo con il necessarioAWS Identity and Access Managementpolitica (IAM) perCloudWatchTronchi. Se crei un flusso di lavoro standard o un flusso di lavoro rapido utilizzando l'API, la CLI oAWS CloudFormation, Step Functions non abilita la registrazione per impostazione predefinita e dovrai assicurarti che il tuo ruolo disponga delle autorizzazioni necessarie.

Per ogni esecuzione avviata dalla console, Step Functions fornisce un collegamento aCloudWatchRegistri, configurati con il filtro corretto per recuperare gli eventi di registro specifici per quell'esecuzione.

Per configurare la registrazione, puoi passare ilLoggingConfigurationparametro quando si utilizzaCreateStateMachineoUpdateStateMachine. Puoi analizzare ulteriormente i tuoi dati inCloudWatchRegistra utilizzandoCloudWatchRegistra approfondimenti. Per ulteriori informazioni, vedereAnalisi dei dati di registro conCloudWatchLogs Insights.

CloudWatchRegistra i payload

Gli eventi della cronologia di esecuzione possono contenere proprietà di input o output nelle loro definizioni. Se l'input o l'output con escape vengono inviati aCloudWatchI registri superano i 248 KB, verranno troncati a causa diCloudWatchRegistra le quote.

- È possibile determinare se un carico utile è stato troncato esaminando il`inputDetails`o`outputDetails`proprietà. Per ulteriori informazioni, vedere`HistoryEventExecutionDataDetails`Tipo di dati.
- Per i flussi di lavoro standard, puoi visualizzare la cronologia completa delle esecuzioni utilizzando`GetExecutionHistory`.
- `GetExecutionHistory`non è disponibile per Express Workflow. Se desideri visualizzare l'input e l'output completi, puoi utilizzare gli ARN Amazon S3. Per ulteriori informazioni, consulta the section called “Usa gli ARN Amazon S3 invece di trasferire payload di grandi dimensioni”.

Policy IAM per l'accesso aCloudWatchRegistri

Dovrai anche configurare il ruolo IAM di esecuzione della tua macchina di stato per avere l'autorizzazione appropriata per accedereCloudWatchRegistri come illustrato nell'esempio seguente.

Esempio di policy IAM

Di seguito è riportata una policy di esempio che è possibile utilizzare per configurare le autorizzazioni. Come illustrato nell'esempio seguente, è necessario specificare*nel`Resource`campo perchéCloudWatchAzioni API, ad esempioCreateLogDeliveryeDescribeLogGroups, non supportaTipi di risorse definiti daAmazon CloudWatch Logs. Per ulteriori informazioni, vedereAzioni definite daAmazon CloudWatch Logs.

- Per informazioni suCloudWatchrisorse, vediCloudWatch Logsrisorse e operazioninelAmazonCloudWatchGuida per l'utente.
- Per informazioni sulle autorizzazioni, è necessario configurare l'invio dei log aCloudWatchRegistri, vediAutorizzazioni utentenella sezione intitolataRegistri inviati aCloudWatch 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": "*"
        }
    ]
}
```

Impossibile accedere alCloudWatchRegistri

Se non riesci ad accedere alCloudWatchRegistri, assicurati di aver fatto quanto segue:

1. Hai configurato il ruolo IAM di esecuzione della tua macchina a stato per avere l'autorizzazione appropriata per accedereCloudWatchTronchi.

   Se stai usando ilCreateStateMachineoUpdateStateMachinerichieste, assicurati di aver specificato il ruolo IAM nel`roleArn`parametro contenente le autorizzazioni come mostrato nelesempio precedente.

2. Controllato ilCloudWatchLa politica delle risorse dei registri non supera il limite di 5120 caratteri perCloudWatchRegistra le politiche sulle risorse.

   Se hai superato il limite di caratteri, rimuovi le autorizzazioni non necessarie dalCloudWatchRegistra la politica delle risorse o prefissa il nome del gruppo di log con`/aws/vendedlogs`, che concederà le autorizzazioni al gruppo di log senza aggiungere altri caratteri alla politica delle risorse. Quando si crea un gruppo di log nella console Step Functions, i nomi dei gruppi di log sono preceduti da`/aws/vendedlogs/states`. Per ulteriori informazioni, consulta Limitazioni relative alle dimensioni della politica delle risorse di Amazon CloudWatch Logs.

Livelli di log

È possibile scegliere tra `OFF`, `ALL`, `ERROR` o `FATAL`. Nessun log dei tipi di evento quando impostato su `OFF` e tutti i tipi di evento quando impostato su `ALL`. Per `ERROR` e `FATAL`, consulta la tabella seguente.

Per ulteriori informazioni sui dati di esecuzione visualizzati per le esecuzioni di Express Workflow basate su questiLivelli di registro, vediEsecuzioni Standard ed Express Workflow nella console.

Tipo di evento	**ALL**	**ERROR**	**FATAL**	**OFF**
ChoiceStateEntered	✓			
ChoiceStateExited	✓			
ExecutionAborted	✓	✓	✓	
ExecutionFailed	✓	✓	✓	
ExecutionStarted	✓			
ExecutionSucceeded	✓			
ExecutionTimedOut	✓	✓	✓	
FailStateEntered	✓	✓		
LambdaFunctionFailed	✓	✓		
LambdaFunctionScheduled	✓			

Tipo di evento	**ALL**	**ERROR**	**FATAL**	**OFF**
LambdaFunctionScheduleFailed	✓	✓		
LambdaFunctionStarted	✓			
LambdaFunctionStartFailed	✓	✓		
LambdaFunctionSucceeded	✓			
LambdaFunctionTimedOut	✓	✓		
MapIterationAborted	✓	✓		
MapIterationFailed	✓	✓		
MapIterationStarted	✓			
MapIterationSucceeded	✓			
MapRunAborted	✓	✓		
MapRunFailed	✓	✓		
MapStateAborted	✓	✓		
MapStateEntered	✓			

Tipo di evento	ALL	ERROR	FATAL	OFF
MapStateExited	✓			
MapStateFailed	✓	✓		
MapStateStarted	✓			
MapStateSucceeded	✓			
ParallelStateAborted	✓	✓		
ParallelStateEntered	✓			
ParallelStateExited	✓			
ParallelStateFailed	✓	✓		
ParallelStateStarted	✓			
ParallelStateSucceeded	✓			
PassStateEntered	✓			
PassStateExited	✓			
SucceedStateEntered	✓			
SucceedStateExited	✓			

Tipo di evento	ALL	ERROR	FATAL	OFF
TaskFailed	✓	✓		
TaskScheduled	✓			
TaskStarted	✓			
TaskStartFailed	✓	✓		
TaskState Aborted	✓	✓		
TaskState Entered	✓			
TaskStateExited	✓			
TaskSubmitFailed	✓	✓		
TaskSubmitted	✓			
TaskSucceeded	✓			
TaskTimedOut	✓	✓		
WaitState Aborted	✓	✓		
WaitState Entered	✓			
WaitStateExited	✓			

AWS X-Ray e Step Functions

È possibile utilizzarli AWS X-Rayper visualizzare i componenti della macchina a stati, identificare i punti deboli delle prestazioni e risolvere le richieste che hanno provocato un errore. La macchina

a stati invia i dati di traccia a X-Ray e X-Ray elabora i dati per generare una mappa dei servizi e riepiloghi delle tracce ricercabili.

Con X-Ray abilitato per la tua macchina a stati, puoi tracciare le richieste man mano che vengono eseguite in Step Functions, in tutte le AWS regioni in cui è disponibile X-Ray. Questo offre una panoramica dettagliata di un'intera richiesta Step Functions. Step Functions invierà tracce a X-Ray per l'esecuzione di macchine a stati, anche quando un ID di traccia non viene passato da un servizio upstream. È possibile utilizzare una mappa dei servizi X-Ray per visualizzare la latenza di una richiesta, inclusi tutti AWS i servizi integrati con X-Ray. È inoltre possibile configurare le regole di campionamento per indicare a X-Ray quali richieste registrare e a quali frequenze di campionamento, in base ai criteri specificati.

Quando X-Ray non è abilitato per la macchina a stati e un servizio upstream non trasmette un ID di traccia, Step Functions non invierà tracce a X-Ray per l'esecuzione di macchine a stati. Tuttavia, se un trace ID viene passato da un servizio upstream, Step Functions invierà quindi tracce a X-Ray per l'esecuzione di macchine a stati.

È possibile utilizzarlo AWS X-Ray con Step Functions nelle regioni in cui entrambi sono supportati. Consulta le pagine degli endpoint e delle quote di Step Functions e X-Ray per informazioni sul supporto regionale per X-Ray e Step Functions.

> **Quote combinate X-Ray e Step Functions**
>
> È possibile aggiungere dati a una traccia per un massimo di sette giorni e interrogare i dati di traccia risalenti a trenta giorni, il periodo di tempo in cui X-Ray archivia i dati di traccia. Le tue tracce saranno soggette a quote di raggi X.. Oltre ad altre quote, X-Ray fornisce una dimensione di traccia minima garantita di 100 KB per le macchine a stati Step Functions. Se a X-Ray vengono forniti più di 100 KB di dati di traccia, la traccia potrebbe congelarsi. Consulta la sezione Quote di servizio della pagina X-Ray Endpoint e quote per ulteriori informazioni su altre quote per X-Ray.

> **Important**
>
> Step Functions non supporta il tracciamento a raggi X per le esecuzioni di workflow secondarie avviate da uno stato di Distributed Map perché è facile superare il limite di dimensione del documento Trace per tali esecuzioni.

Argomenti

- Installazione e configurazione
- Concetti
- Integrazioni dei servizi Step Functions e X-Ray
- Visualizzazione della console X-Ray
- Visualizzazione delle informazioni di tracciamento X-Ray per Step Functions
- Tracce
- Mappa servizio
- Segmenti e sottosegmenti
- Analisi
- Configurazione
- Cosa succede se non ci sono dati nella mappa di traccia o nella mappa dei servizi?

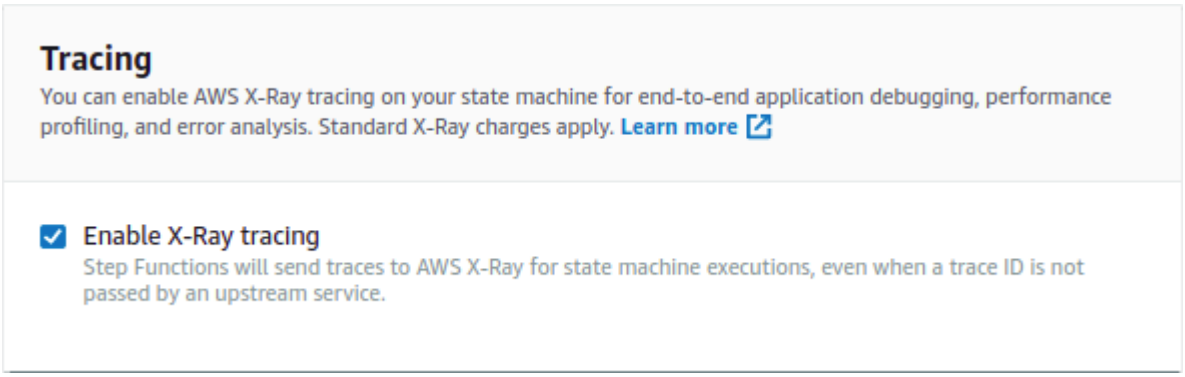
Installazione e configurazione

Abilita il tracciamento a raggi X durante la creazione di una macchina a stati

È possibile abilitare il tracciamento a raggi X durante la creazione di una nuova macchina a stati selezionando Abilita il tracciamento a raggi X nella pagina Specificare i dettagli.

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Nella pagina Scegli il metodo di creazione, scegli l'opzione appropriata per creare la tua macchina a stati. Se si sceglie Esegui un progetto di esempio, non è possibile abilitare il tracciamento a raggi X durante la creazione della macchina a stati e sarà necessario abilitare il tracciamento a raggi X dopo la creazione della macchina a stati. Per ulteriori informazioni sull'attivazione di X-Ray in una macchina a stati esistente, vedere. Abilita X-Ray in una macchina a stati esistente

   Seleziona Avanti.
3. Nella pagina Specificare i dettagli, configura la tua macchina a stati.
4. Scegliete Abilita tracciamento X-Ray.



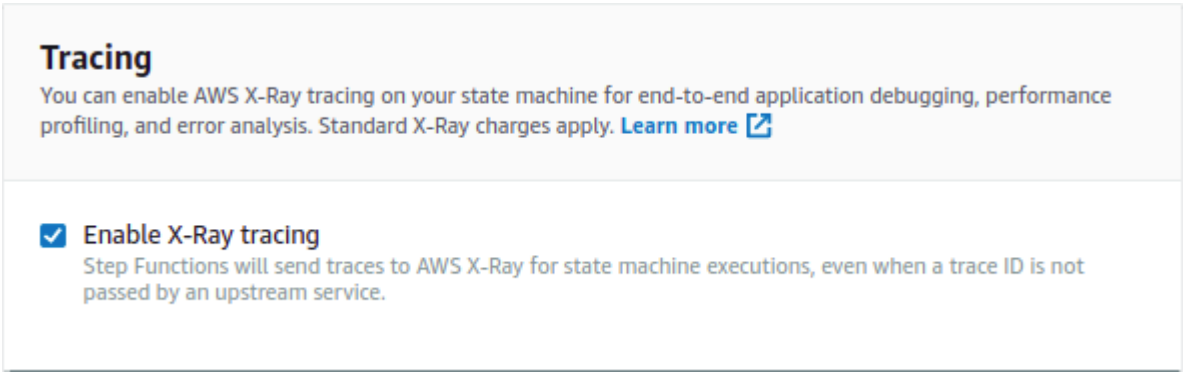
La macchina a stati Step Functions ora invierà tracce a X-Ray per l'esecuzione di macchine a stati.

> **Note**
>
> Se scegli di utilizzare un ruolo IAM esistente, devi assicurarti che le scritture X-Ray siano consentite. Per ulteriori informazioni sulle autorizzazioni necessarie, consulta le politiche IAM per X-Ray.

Abilita X-Ray in una macchina a stati esistente

Per abilitare X-Ray in una macchina a stati esistente:

1. Nella console Step Functions, seleziona la macchina a stati per la quale desideri abilitare il tracciamento.
2. Scegli Modifica.
3. Scegliete Abilita tracciamento X-Ray.



Riceverai una notifica che ti comunica che potresti dover apportare ulteriori modifiche.

> **Note**
>
> Quando abiliti X-Ray per una macchina a stati esistente, devi assicurarti di disporre di una policy IAM che conceda autorizzazioni sufficienti a X-Ray per eseguire tracce. È possibile aggiungerne uno manualmente o generarne uno. Per ulteriori informazioni, consulta la sezione sulla politica IAM perPolitiche IAM per AWS X-Ray.

4. (Facoltativo) Genera automaticamente un nuovo ruolo per la tua macchina a stati per includere le autorizzazioni X-Ray.
5. Selezionare Salva.

Configurazione del tracciamento X-Ray per Step Functions

Quando si esegue per la prima volta una macchina a stati con il tracciamento a raggi X abilitato, utilizzerà i valori di configurazione predefiniti per il tracciamento a raggi X. AWS X-Ray non raccoglie dati per ogni richiesta inviata a un'applicazione. Raccoglie invece dati per un numero statisticamente significativo di richieste. L'impostazione predefinita prevede la registrazione della prima richiesta ogni secondo e del cinque percento di eventuali richieste aggiuntive. Una richiesta al secondo è la riserva. In questo modo viene registrata almeno una traccia al secondo, purché il servizio soddisfi le richieste. Cinque percento è la percentuale di richieste aggiuntive oltre la dimensione della riserva che vengono campionate.

Per evitare di incorrere in costi del servizio durante l'acquisizione delle nozioni di base, il tasso di campionamento di default è conservativo. È possibile configurare X-Ray per modificare la regola di campionamento predefinita e configurare regole aggiuntive che applicano il campionamento in base alle proprietà del servizio o della richiesta.

Ad esempio, potresti voler disabilitare il campionamento e tracciare tutte le richieste di chiamate che modificano lo stato o la gestione delle transazioni. Account AWS Per le chiamate di sola lettura ad alto volume, come il polling in background, i controlli di integrità o la manutenzione della connessione, è possibile campionare a una frequenza bassa e ottenere comunque dati sufficienti per osservare i problemi che si verificano.

Per configurare una regola di campionamento per la tua macchina a stati:

1. Vai alla console X-Ray.
2. Scegli Campionamento.

3. Per creare una regola, scegliere Create sampling rule (Crea regola di campionamento).

   Per modificare una regola, scegliere il nome di una regola.

   Per eliminare una regola, sceglierne una e utilizzare il menu Actions (Operazioni) per eliminarla.

Alcune parti delle regole di campionamento esistenti, come il nome e la priorità, non possono essere modificate. Invece, aggiungi o clona una regola esistente, apporta le modifiche desiderate, quindi usa la nuova regola.

Per informazioni dettagliate sulle regole di campionamento a raggi X e su come configurare i vari parametri, vedere Configurazione delle regole di campionamento nella console X-Ray.

Integra i servizi upstream

Per integrare l'esecuzione dei flussi di lavoro Step Functions, come i flussi di lavoro Express, Synchronous e Standard, con un servizio upstream è necessario impostare il. `traceHeader` Questa operazione viene eseguita automaticamente se si utilizza un'API HTTP in API Gateway. Tuttavia, se utilizzi una funzione Lambda e/o un SDK, devi impostare tu stesso le chiamate `traceHeader` on the `StartExecution`o `StartSyncExecution`API.

Devi specificare il `traceHeader` formato come. `\p{ASCII}#` Inoltre, per consentire a Step Functions di utilizzare lo stesso ID di traccia, è necessario specificare il formato di`Root={TRACE_ID};Sampled={1 or 0}`. Se utilizzi una funzione Lambda, sostituisci l'`TRACE_ID`ID di traccia nel segmento corrente e imposta il campo Sampled come `1` se la modalità di campionamento fosse vera e `0` se la modalità di campionamento fosse falsa. Fornendo l'ID di traccia in questo formato, otterrai una traccia completa.

Di seguito è riportato un esempio scritto in Python per mostrare come specificare il. `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)
```

Concetti

La console X-Ray

La AWS X-Ray console consente di visualizzare le mappe e le tracce dei servizi per le richieste servite dalle applicazioni. Puoi accedere alla console per visualizzare le informazioni dettagliate raccolte da X-Ray quando è abilitato per la tua macchina a stati.

Visualizzazione della console X-RayPer informazioni su come accedere alla console X-Ray per le esecuzioni delle macchine a stati, vedere.

Per informazioni dettagliate sulla console X-Ray, vedere la documentazione della console X-Ray.

Segmenti, sottosegmenti e tracce

Un segmento registra le informazioni relative a una richiesta inviata alla macchina a stati. Contiene informazioni come il lavoro svolto dalla macchina a stati e può contenere anche sottosegmenti con informazioni sulle chiamate downstream.

Una traccia raccoglie tutti i segmenti generati da una singola richiesta.

Campionamento

Per garantire un tracciamento efficiente e fornire un campione rappresentativo delle richieste servite dall'applicazione, X-Ray applica un algoritmo di campionamento per determinare quali richieste vengono tracciate. Questo può essere modificato modificando le regole di campionamento.

Metriche

Per la vostra macchina a stati, X-Ray misurerà il tempo di invocazione, il tempo di transizione di stato, il tempo di esecuzione complessivo di Step Functions e le variazioni in questo tempo di esecuzione. È possibile accedere a queste informazioni tramite la console X-Ray.

Analisi

La console AWS X-Ray Analytics è uno strumento interattivo per l'interpretazione dei dati di traccia. È possibile definire il set di dati attivo con filtri sempre più granulari facendo clic sui grafici e i pannelli di parametri e campi associati con il set di tracciamenti corrente. Ciò consente di analizzare le prestazioni della macchina a stati e di individuare e identificare rapidamente i problemi di prestazioni.

Per informazioni dettagliate sull'analisi a raggi X, consulta Interagire con la console Analytics AWS X-Ray

Integrazioni dei servizi Step Functions e X-Ray

Alcuni dei AWS servizi che si integrano con Step Functions forniscono l'integrazione AWS X-Ray aggiungendo un'intestazione di tracciamento alle richieste, eseguendo il demone X-Ray o prendendo decisioni di campionamento e caricando i dati di traccia su X-Ray. Altri devono essere strumentati utilizzando l'SDK. AWS X-Ray Alcuni non supportano ancora l'integrazione con i raggi X. L'integrazione X-Ray è necessaria per fornire dati di tracciamento completi quando si utilizza un'integrazione di servizi con Step Functions

Supporto X-Ray nativo

Le integrazioni di servizi con supporto X-Ray nativo includono:

- Amazon Simple Notification Service
- Amazon Simple Queue Service
- AWS Lambda
- AWS Step Functions

Strumentazione richiesta

Integrazioni di servizi che richiedono strumentazione a raggi X:

- Amazon Elastic Container Service
- AWS Batch
- AWS Fargate

Traccia solo lato client

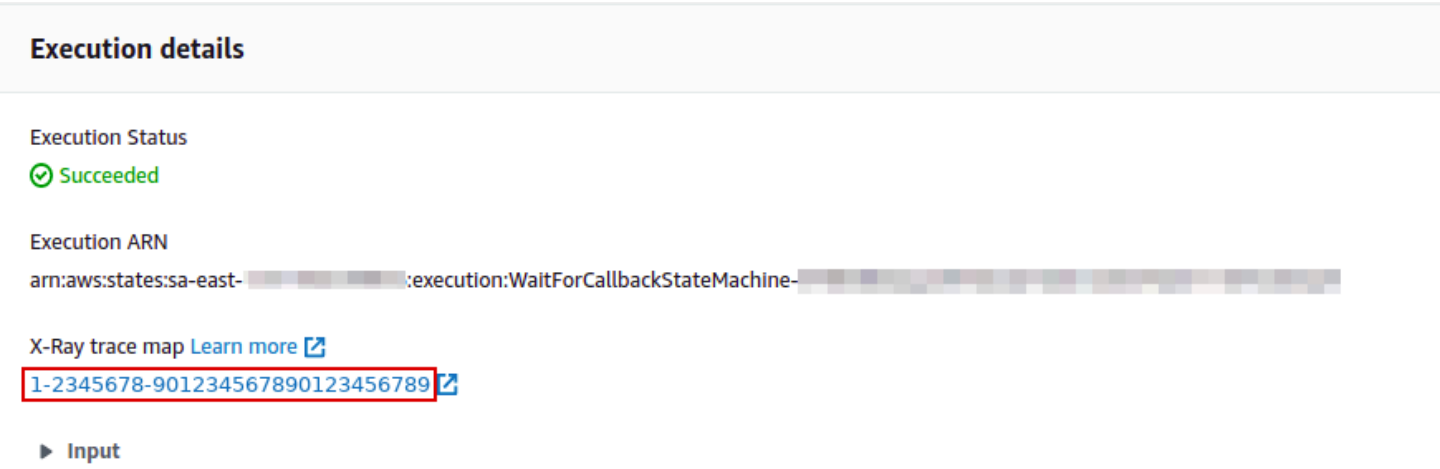
Altre integrazioni di servizi non supportano le tracce X-Ray. Tuttavia, le tracce lato client possono ancora essere raccolte:

- Amazon DynamoDB
- Amazon EMR
- Amazon SageMaker
- AWS CodeBuild
- AWS Glue

Visualizzazione della console X-Ray

X-Ray riceve i dati dai servizi come segmenti. X-Ray raggruppa i segmenti che hanno una richiesta comune in tracce. X-Ray elabora le tracce per generare un grafico di servizio che fornisce una rappresentazione visiva dell'applicazione.

Dopo aver avviato l'esecuzione della macchina a stati, è possibile visualizzare le relative tracce a raggi X scegliendo il collegamento alla mappa di traccia a raggi X nella sezione Dettagli di esecuzione.



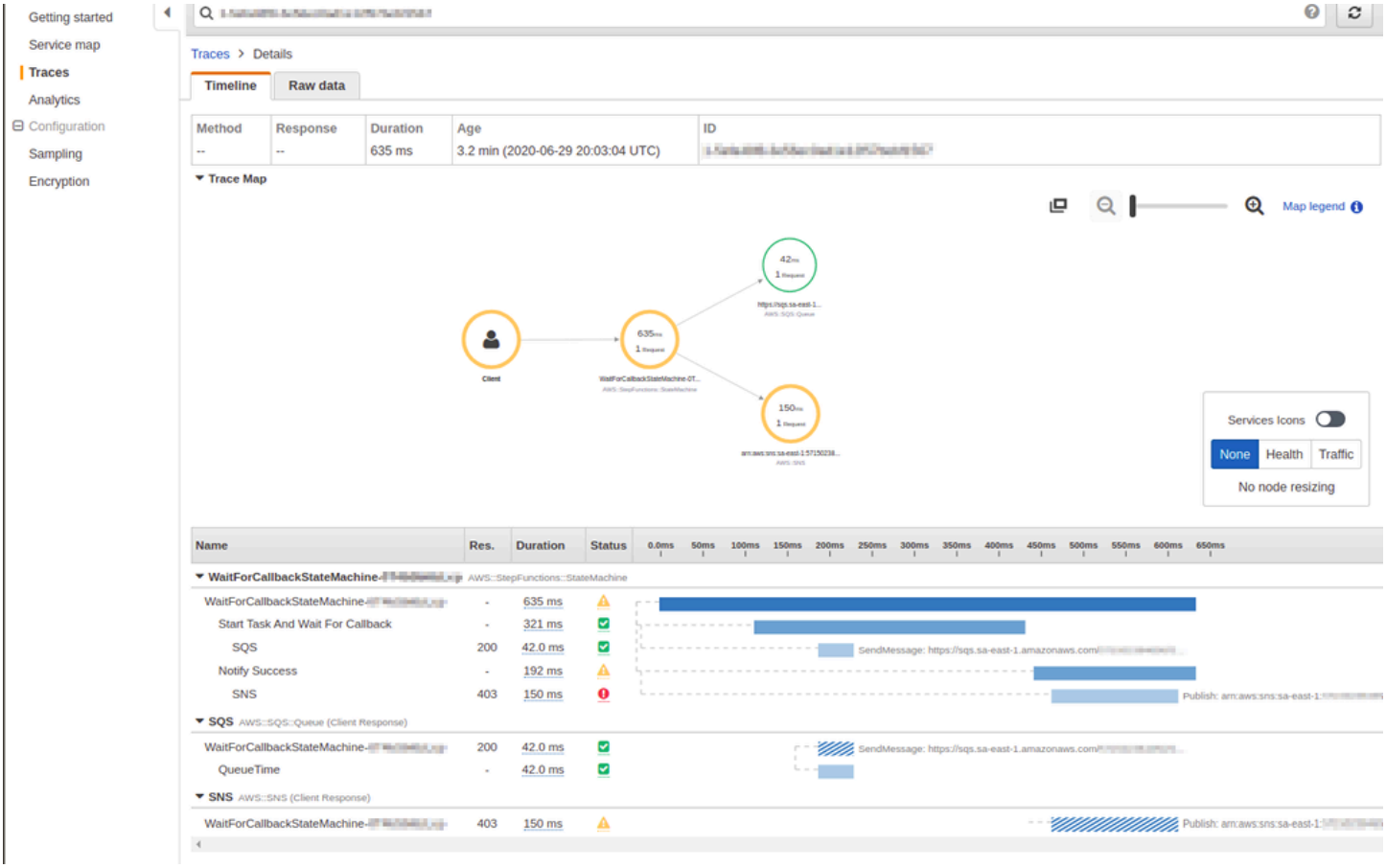
Dopo aver abilitato X-Ray per la tua macchina a stati, puoi visualizzare le informazioni di tracciamento per le sue esecuzioni nella console X-Ray.

Visualizzazione delle informazioni di tracciamento X-Ray per Step Functions

I passaggi seguenti illustrano il tipo di informazioni che è possibile visualizzare nella console dopo aver abilitato X-Ray ed eseguito un'esecuzione. Vengono mostrate le tracce a raggi X per il progetto Esempio di pattern di callback (Amazon SQS, Amazon SNS, Lambda) di esempio.

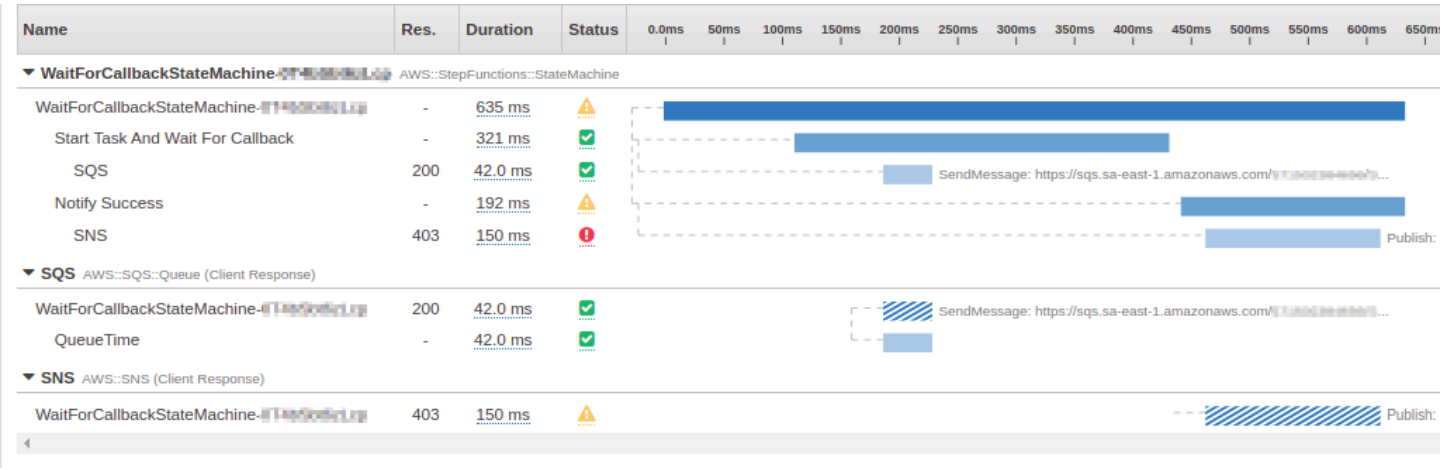
Tracce

Al termine dell'esecuzione, puoi accedere alla console X-Ray, dove vedrai la pagina X-Ray Traces. Viene visualizzata una panoramica della mappa dei servizi e informazioni di traccia e segmento per la macchina a stati.

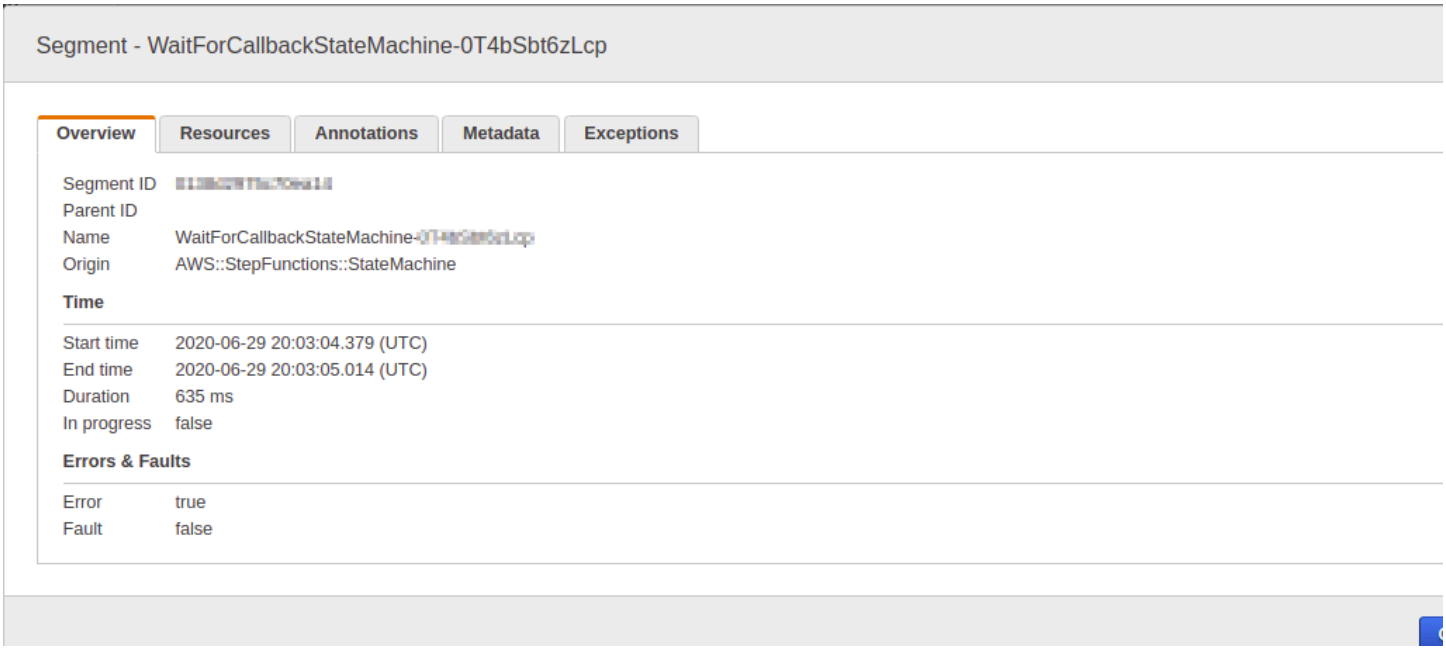


Mappa servizio

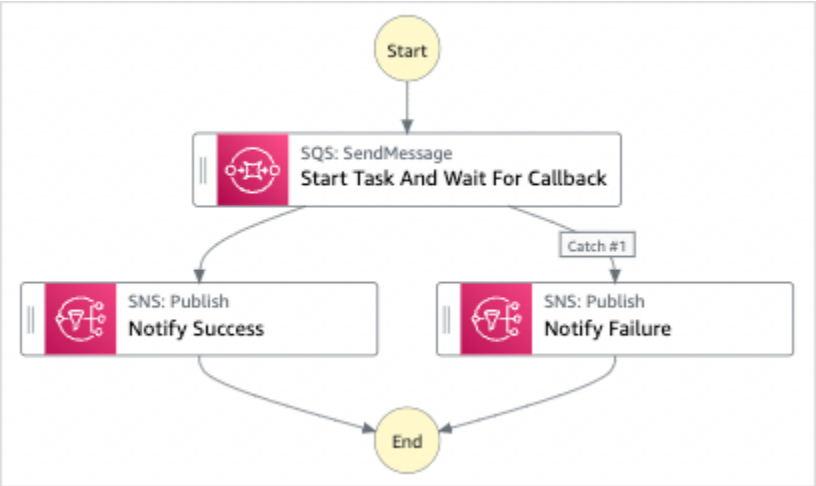
La mappa dei servizi nella console X-Ray consente di identificare i servizi in cui si verificano errori, in cui sono presenti connessioni con latenza elevata o a visualizzare le tracce delle richieste non riuscite.



Sulla mappa di traccia, puoi scegliere un nodo di servizio per visualizzare le richieste per quel nodo o un edge tra due nodi per visualizzare le richieste che hanno percorso quella connessione. Qui, il `WaitForCallBack` nodo è stato selezionato ed è possibile visualizzare informazioni aggiuntive sul suo stato di esecuzione e risposta.



Puoi vedere come la mappa del servizio X-Ray è correlata alla macchina a stati. Esiste un nodo della mappa dei servizi per ogni integrazione di servizi chiamato da Step Functions, a condizione che supporti X-Ray.



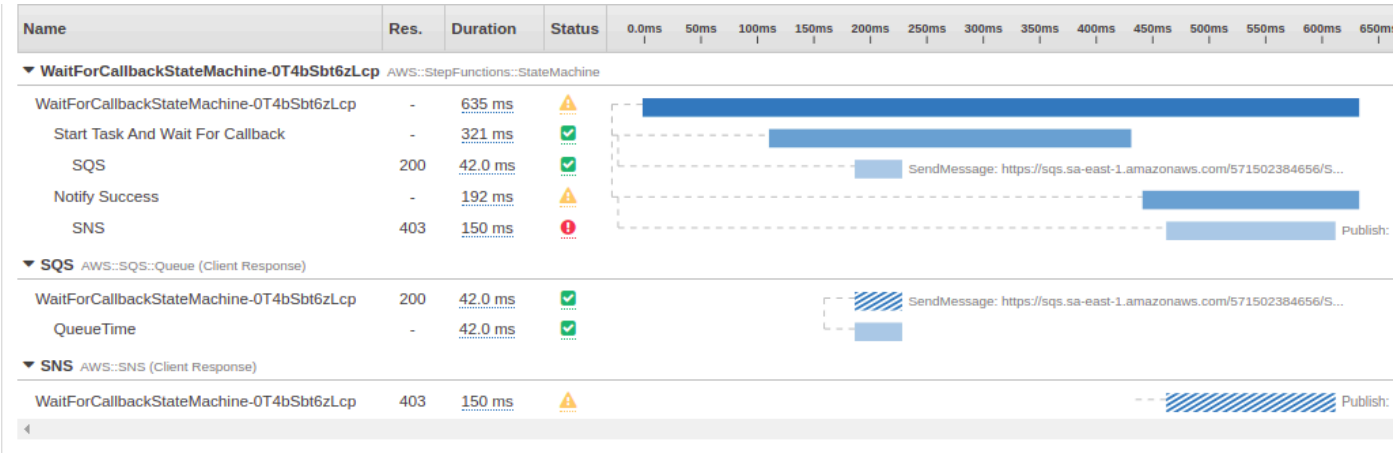
Segmenti e sottosegmenti

Una traccia è una raccolta di segmenti generati da una singola richiesta. Ogni segmento fornisce il nome della risorsa, dettagli sulla richiesta e dettagli sul lavoro svolto. Nella pagina Traces, puoi vedere i segmenti e, se espansi, i relativi sottosegmenti corrispondenti. Puoi scegliere un segmento o un sottosegmento per visualizzarne informazioni dettagliate.

Scegli ciascuna scheda per vedere come vengono visualizzate le informazioni per segmenti e sottosegmenti.

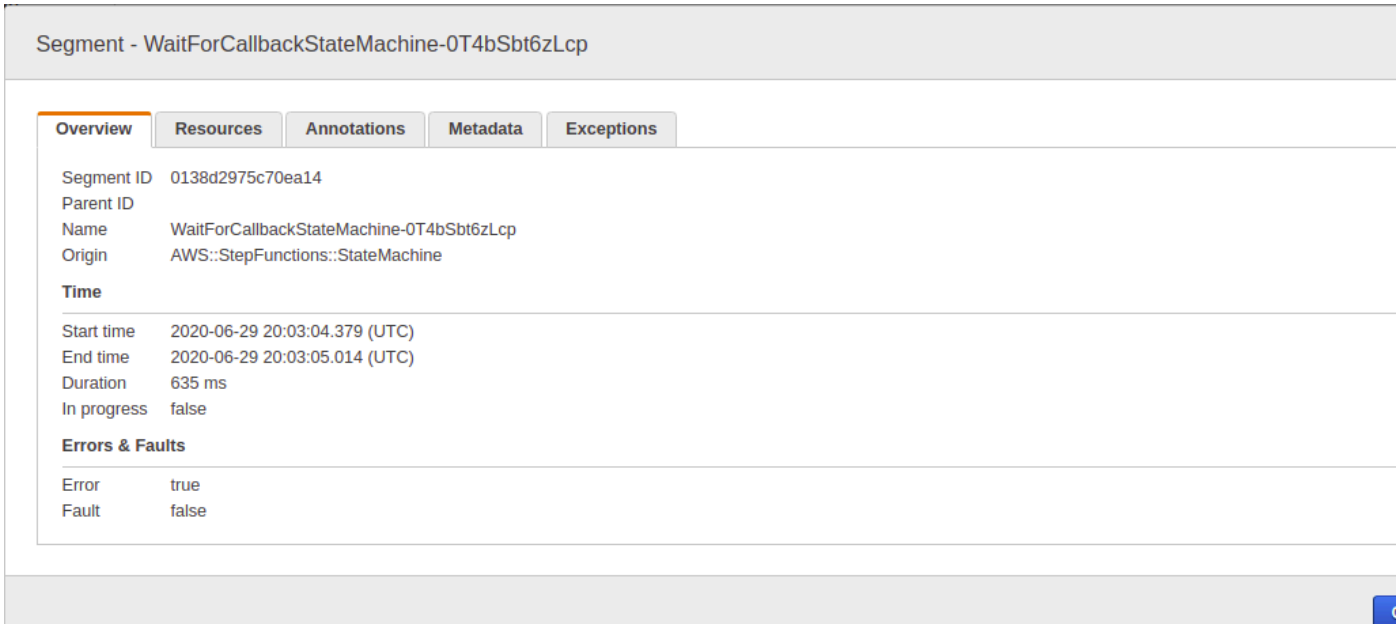
Overview of Segments

Una panoramica dei segmenti e dei sottosegmenti per questa macchina a stati. C'è un segmento diverso per ogni nodo sulla mappa dei servizi.



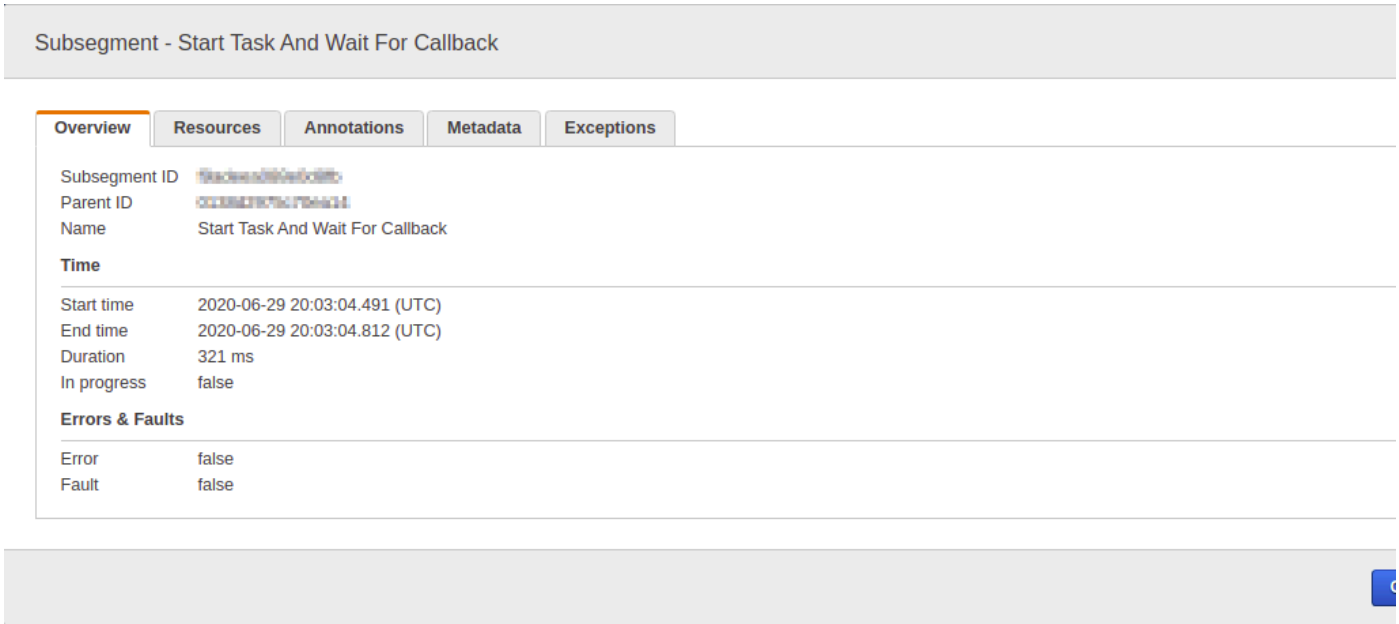
View segment detail

La scelta di un segmento fornisce il nome della risorsa, i dettagli sulla richiesta e i dettagli sul lavoro svolto.



View subsegment detail

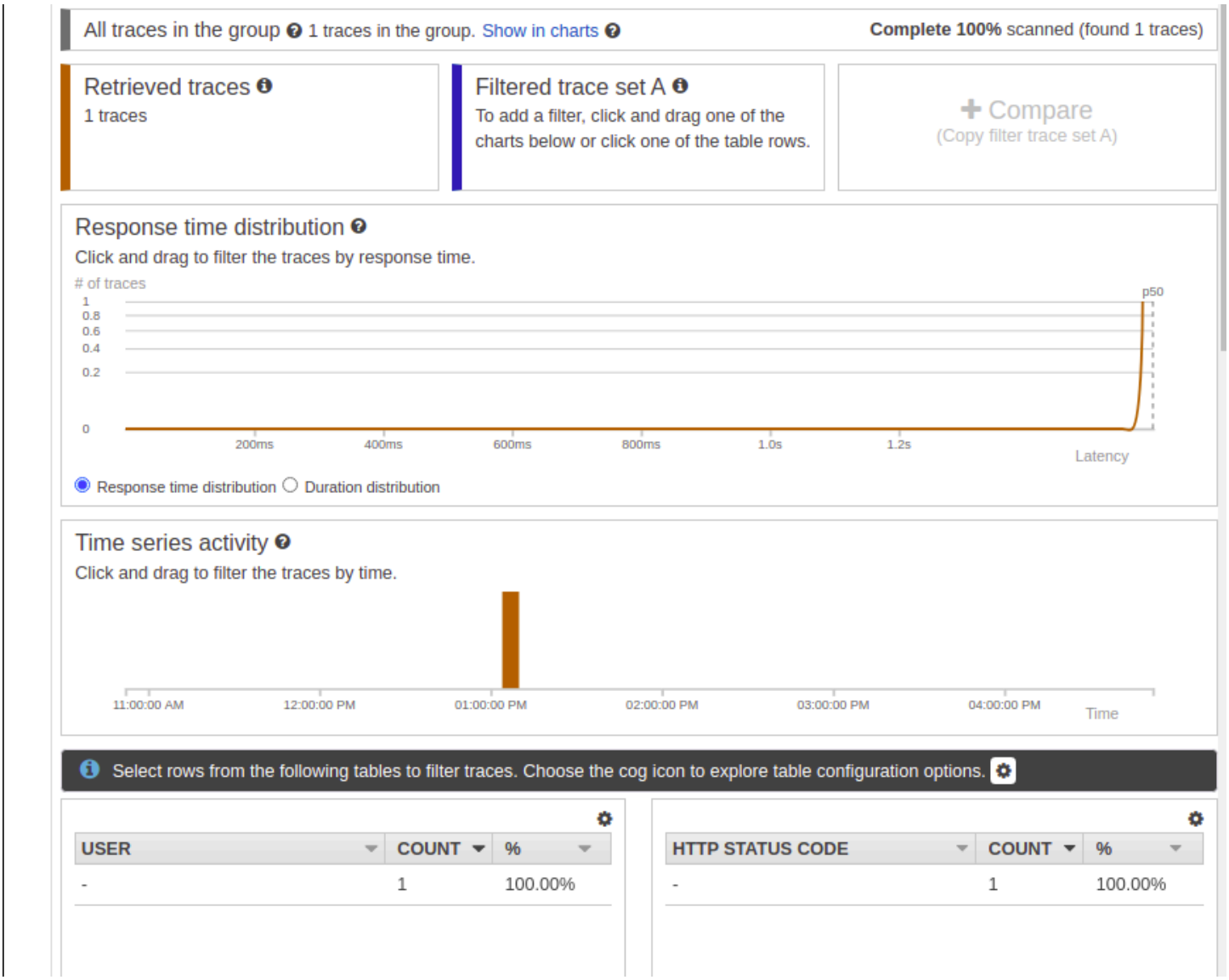
Un segmento può suddividere i dati sull'attività eseguita in sottosegmenti. La scelta di un sottosegmento consente di visualizzare informazioni e dettagli sulla tempistica più granulari. Un sottosegmento può contenere dettagli aggiuntivi su una chiamata a un AWS servizio, un'API HTTP esterna o un database SQL.



Analisi

La console AWS X-Ray Analytics è uno strumento interattivo per l'interpretazione dei dati di traccia. Puoi usarlo per comprendere più facilmente le prestazioni della tua macchina a stati. La console consente di esplorare, analizzare e visualizzare i tracciamenti tramite tempo di risposta interattivo e grafici di serie temporali. Questo può aiutarti a individuare rapidamente i problemi di prestazioni e latenza.

È possibile definire il set di dati attivo con filtri sempre più granulari facendo clic sui grafici e i pannelli di parametri e campi associati con il set di tracciamenti corrente.

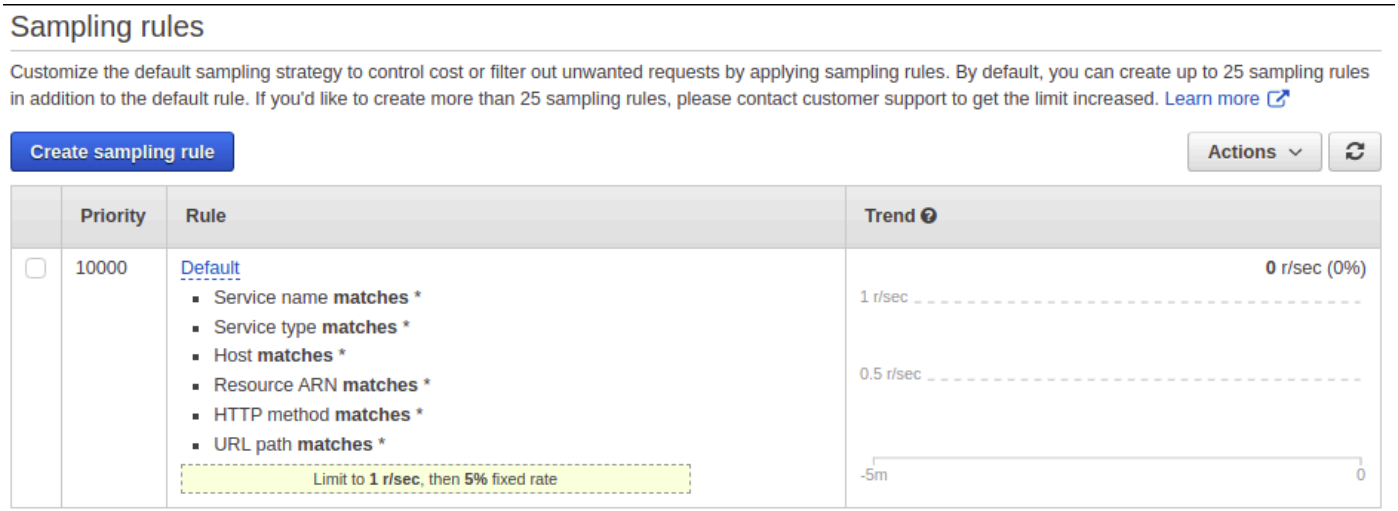


Configurazione

È possibile configurare le opzioni di campionamento e crittografia dalla console X-Ray.

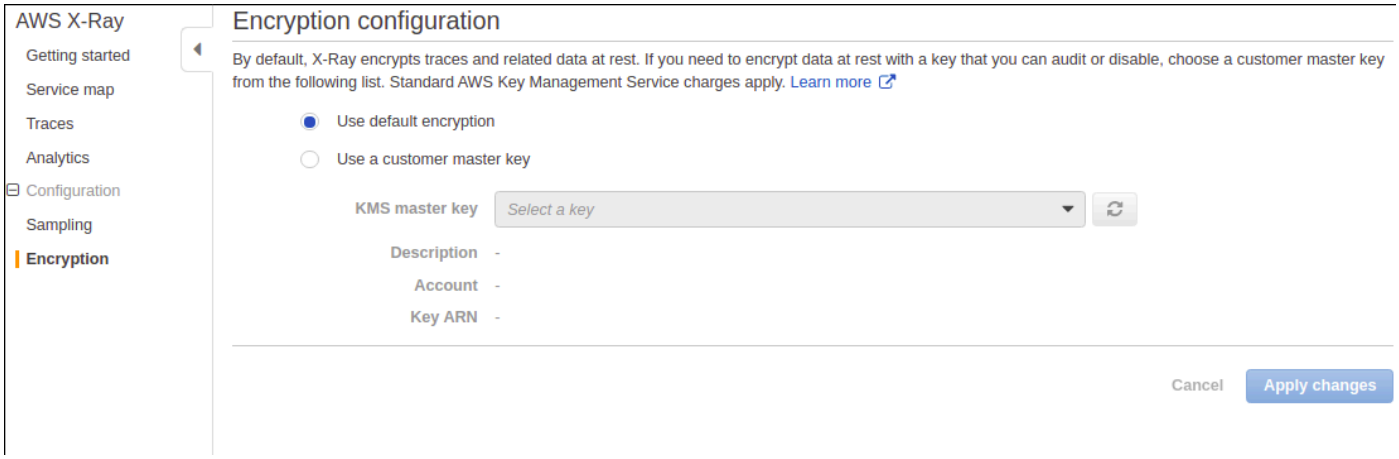
Sampling

Scegliete Campionamento per visualizzare i dettagli sulla frequenza di campionamento e sulla configurazione. È possibile modificare le regole di campionamento per controllare la quantità di dati registrati e modificare il comportamento di campionamento in base alle proprie esigenze specifiche.



Encryption

Scegliete Crittografia per modificare le impostazioni di crittografia. È possibile utilizzare l'impostazione predefinita, in cui X-Ray crittografa le tracce e la data a riposo oppure, se necessario, è possibile scegliere una chiave master del cliente. In quest'ultimo caso si applicano le AWS KMStariffe standard.



Cosa succede se non ci sono dati nella mappa di traccia o nella mappa dei servizi?

Se hai abilitato X-Ray, ma non riesci a visualizzare alcun dato nella console X-Ray, verifica che:

- I ruoli IAM sono configurati correttamente per consentire la scrittura su X-Ray.
- Le regole di campionamento consentono il campionamento dei dati.
- Poiché può esserci un breve ritardo prima che vengano applicati i ruoli IAM appena creati o modificati, ricontrolla la traccia o le mappe dei servizi dopo alcuni minuti.
- Se vedi Data non trovata nel pannello X-Ray Traces, controlla le impostazioni del tuo account IAM e assicurati che AWS Security Token Service sia abilitato per la regione desiderata. Per ulteriori informazioni, consulta Attivazione e disattivazione AWS STSRegione AWS in una delle guide per l'utente di IAM.

Uso di Notifiche all'utente AWS con AWS Step Functions

Puoi utilizzarlo Notifiche all'utente AWSper configurare i canali di distribuzione per ricevere notifiche sugli AWS Step Functions eventi. L'utente riceverà una notifica quando un evento corrisponde a una regola specificata. È possibile ricevere notifiche per gli eventi tramite più canali, tra cui e-mail, notifiche chat AWS Chatbot o notifiche push AWS Console Mobile Application. Puoi anche visualizzare le notifiche nel Centro notifiche della console. Notifiche all'utente supporta l'aggregazione, che può ridurre il numero di notifiche ricevute durante eventi specifici.

Sicurezza in AWS Step Functions

Questa sezione fornisce informazioni sulla AWS Step Functions sicurezza e l'autenticazione.

Step Functions utilizza IAM per controllare l'accesso ad altri AWS servizi e risorse. Per una panoramica sul funzionamento di IAM, vedi Panoramica della gestione degli accessi nella Guida per l'utente di IAM. Per una panoramica delle credenziali di sicurezza, consultare Credenziali di sicurezza AWS in Riferimenti generali di Amazon Web Services.

Protezione dei dati in AWS Step Functions

Il modello di responsabilità AWS condivisa modello di di si applica alla protezione dei dati in AWS Step Functions. Come descritto in questo modello, AWS è responsabile della protezione dell'infrastruttura globale che gestisce tutti i Cloud AWS. L'utente è responsabile del controllo dei contenuti ospitati su questa infrastruttura. L'utente è inoltre responsabile della configurazione della protezione e delle attività di gestione per i Servizi AWS utilizzati. Per ulteriori informazioni sulla privacy dei dati, vedi le Domande frequenti sulla privacy dei dati. Per informazioni sulla protezione dei dati in Europa, consulta il post del blog relativo al Modello di responsabilità condivisa AWS e GDPR nel Blog sulla sicurezza AWS .

Ai fini della protezione dei dati, consigliamo di proteggere Account AWS le credenziali e configurare i singoli utenti con AWS IAM Identity Center or AWS Identity and Access Management (IAM). In tal modo, a ogni utente verranno assegnate solo le autorizzazioni necessarie per svolgere i suoi compiti. Ti suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- Usa SSL/TLS per comunicare con le risorse. AWS È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Configura l'API e la registrazione delle attività degli utenti con. AWS CloudTrail
- Utilizza soluzioni di AWS crittografia, insieme a tutti i controlli di sicurezza predefiniti all'interno Servizi AWS.
- Utilizza i servizi di sicurezza gestiti avanzati, come Amazon Macie, che aiutano a individuare e proteggere i dati sensibili archiviati in Amazon S3.
- Se hai bisogno di moduli crittografici convalidati FIPS 140-2 per l'accesso AWS tramite un'interfaccia a riga di comando o un'API, utilizza un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il Federal Information Processing Standard (FIPS) 140-2.

Ti consigliamo vivamente di non inserire mai informazioni riservate o sensibili, ad esempio gli indirizzi e-mail dei clienti, nei tag o nei campi di testo in formato libero, ad esempio nel campo Nome. Ciò include quando lavori con Step Functions o altro Servizi AWS utilizzando la console, l'API o AWS gli SDK. AWS CLI I dati inseriti nei tag o nei campi di testo in formato libero utilizzati per i nomi possono essere utilizzati per i la fatturazione o i log di diagnostica. Quando fornisci un URL a un server esterno, ti suggeriamo vivamente di non includere informazioni sulle credenziali nell'URL per convalidare la tua richiesta al server.

Crittografia in AWS Step Functions

Crittografia dei dati inattivi

Step Functions crittografa sempre i dati a riposo. I dati in ingresso AWS Step Functions vengono crittografati quando sono inattivi utilizzando una crittografia trasparente lato server. Questo consente di ridurre gli oneri operativi e la complessità associati alla protezione dei dati sensibili. La crittografia dei dati inattivi consente di creare applicazioni sicure che rispettano rigorosi requisiti normativi e di conformità per la crittografia.

Crittografia in transito

Step Functions crittografa i dati in transito tra il servizio e altri AWS servizi integrati (vediUtilizzo AWS Step Functions con altri servizi). Tutti i dati che passano tra Step Functions e i servizi integrati sono crittografati utilizzando Transport Layer Security (TLS).

Identity and Access Management in AWS Step Functions

L'accesso a AWS Step Functions richiede credenziali che AWS possono essere utilizzate per autenticare le richieste. Queste credenziali devono disporre delle autorizzazioni per accedere alle AWS risorse, ad esempio recuperare i dati degli eventi da altre risorse. AWS Le seguenti sezioni forniscono dettagli su come utilizzare AWS Identity and Access Management (IAM) e Step Functions per proteggere le risorse controllando chi può accedervi.

AWS Identity and Access Management (IAM) è un software Servizio AWS che aiuta un amministratore a controllare in modo sicuro l'accesso alle AWS risorse. Gli amministratori IAM controllano chi può essere autenticato (effettuato l'accesso) e autorizzato (dispone delle autorizzazioni) a utilizzare le risorse Step Functions. IAM è uno Servizio AWS strumento che puoi utilizzare senza costi aggiuntivi.

Destinatari

Il modo in cui usi AWS Identity and Access Management (IAM) varia a seconda del lavoro svolto in Step Functions.

Utente del servizio: se utilizzi il servizio Step Functions per svolgere il tuo lavoro, l'amministratore ti fornisce le credenziali e le autorizzazioni necessarie. Man mano che utilizzi più funzioni di Step Functions per svolgere il tuo lavoro, potresti aver bisogno di autorizzazioni aggiuntive. La comprensione della gestione dell'accesso ti consente di richiedere le autorizzazioni corrette all'amministratore. Se non riesci ad accedere a una funzionalità di Step Functions, vediRisoluzione dei problemi di AWS Step Functions identità e accesso.

Amministratore del servizio: se sei responsabile delle risorse Step Functions della tua azienda, probabilmente hai pieno accesso a Step Functions. Il tuo compito è determinare a quali funzionalità e risorse di Step Functions gli utenti del servizio devono accedere. Devi inviare le richieste all'amministratore IAM per cambiare le autorizzazioni degli utenti del servizio. Esamina le informazioni contenute in questa pagina per comprendere i concetti di base relativi a IAM. Per saperne di più su come la tua azienda può utilizzare IAM con Step Functions, consultaCome AWS Step Functions funziona con IAM.

Amministratore IAM: se sei un amministratore IAM, potresti voler conoscere i dettagli su come scrivere policy per gestire l'accesso a Step Functions. Per visualizzare esempi di policy basate sull'identità di Step Functions che puoi utilizzare in IAM, consulta. Esempi di policy basate sull'identità per AWS Step Functions

Autenticazione con identità

L'autenticazione è il modo in cui accedi AWS utilizzando le tue credenziali di identità. Devi essere autenticato (aver effettuato l' Utente root dell'account AWS accesso AWS) come utente IAM o assumendo un ruolo IAM.

Puoi accedere AWS come identità federata utilizzando le credenziali fornite tramite una fonte di identità. AWS IAM Identity Center Gli utenti (IAM Identity Center), l'autenticazione Single Sign-On della tua azienda e le tue credenziali di Google o Facebook sono esempi di identità federate. Se accedi come identità federata, l'amministratore ha configurato in precedenza la federazione delle identità utilizzando i ruoli IAM. Quando accedi AWS utilizzando la federazione, assumi indirettamente un ruolo.

A seconda del tipo di utente, puoi accedere al AWS Management Console o al portale di AWS accesso. Per ulteriori informazioni sull'accesso a AWS, vedi Come accedere al tuo Account AWS nella Guida per l'Accedi ad AWS utente.

Se accedi a AWS livello di codice, AWS fornisce un kit di sviluppo software (SDK) e un'interfaccia a riga di comando (CLI) per firmare crittograficamente le tue richieste utilizzando le tue credenziali. Se non utilizzi AWS strumenti, devi firmare tu stesso le richieste. Per ulteriori informazioni sull'utilizzo del metodo consigliato per firmare autonomamente le richieste, consulta Signing AWS API request nella IAM User Guide.

A prescindere dal metodo di autenticazione utilizzato, potrebbe essere necessario specificare ulteriori informazioni sulla sicurezza. Ad esempio, ti AWS consiglia di utilizzare l'autenticazione a più fattori (MFA) per aumentare la sicurezza del tuo account. Per ulteriori informazioni, consulta Autenticazione a più fattori nella Guida per l'utente di AWS IAM Identity Center e Utilizzo dell'autenticazione a più fattori (MFA) in AWS nella Guida per l'utente di IAM.

Account AWS utente root

Quando si crea un account Account AWS, si inizia con un'identità di accesso che ha accesso completo a tutte Servizi AWS le risorse dell'account. Questa identità è denominata utente Account AWS root ed è accessibile effettuando l'accesso con l'indirizzo e-mail e la password utilizzati per creare l'account. Si consiglia vivamente di non utilizzare l'utente root per le attività quotidiane. Conserva le credenziali dell'utente root e utilizzarle per eseguire le operazioni che solo l'utente root può eseguire. Per un elenco completo delle attività che richiedono l'accesso come utente root, consulta la sezione Attività che richiedono le credenziali dell'utente root nella Guida per l'utente di IAM.

Identità federata

Come procedura consigliata, richiedi agli utenti umani, compresi gli utenti che richiedono l'accesso come amministratore, di utilizzare la federazione con un provider di identità per accedere Servizi AWS utilizzando credenziali temporanee.

Un'identità federata è un utente dell'elenco utenti aziendale, un provider di identità Web AWS Directory Service, la directory Identity Center o qualsiasi utente che accede Servizi AWS utilizzando credenziali fornite tramite un'origine di identità. Quando le identità federate accedono Account AWS, assumono ruoli e i ruoli forniscono credenziali temporanee.

Per la gestione centralizzata degli accessi, consigliamo di utilizzare AWS IAM Identity Center. Puoi creare utenti e gruppi in IAM Identity Center oppure puoi connetterti e sincronizzarti con un set di

utenti e gruppi nella tua fonte di identità per utilizzarli su tutte le tue applicazioni. Account AWS Per ulteriori informazioni sul Centro identità IAM, consulta Cos'è Centro identità IAM? nella Guida per l'utente di AWS IAM Identity Center .

Utenti e gruppi IAM

Un utente IAM è un'identità interna Account AWS che dispone di autorizzazioni specifiche per una singola persona o applicazione. Ove possibile, consigliamo di fare affidamento a credenziali temporanee invece di creare utenti IAM con credenziali a lungo termine come le password e le chiavi di accesso. Tuttavia, per casi d'uso specifici che richiedono credenziali a lungo termine con utenti IAM, si consiglia di ruotare le chiavi di accesso. Per ulteriori informazioni, consulta la pagina Rotazione periodica delle chiavi di accesso per casi d'uso che richiedono credenziali a lungo termine nella Guida per l'utente di IAM.

Un gruppo IAM è un'identità che specifica un insieme di utenti IAM. Non è possibile eseguire l'accesso come gruppo. È possibile utilizzare gruppi per specificare le autorizzazioni per più utenti alla volta. I gruppi semplificano la gestione delle autorizzazioni per set di utenti di grandi dimensioni. Ad esempio, è possibile avere un gruppo denominato Amministratori IAM e concedere a tale gruppo le autorizzazioni per amministrare le risorse IAM.

Gli utenti sono diversi dai ruoli. Un utente è associato in modo univoco a una persona o un'applicazione, mentre un ruolo è destinato a essere assunto da chiunque ne abbia bisogno. Gli utenti dispongono di credenziali a lungo termine permanenti, mentre i ruoli forniscono credenziali temporanee. Per ulteriori informazioni, consulta Quando creare un utente IAM (invece di un ruolo) nella Guida per l'utente di IAM.

Ruoli IAM

Un ruolo IAM è un'identità interna all'utente Account AWS che dispone di autorizzazioni specifiche. È simile a un utente IAM, ma non è associato a una persona specifica. Puoi assumere temporaneamente un ruolo IAM in AWS Management Console cambiando ruolo. Puoi assumere un ruolo chiamando un'operazione AWS CLI o AWS API o utilizzando un URL personalizzato. Per ulteriori informazioni sui metodi per l'utilizzo dei ruoli, consulta Utilizzo di ruoli IAM nella Guida per l'utente di IAM.

I ruoli IAM con credenziali temporanee sono utili nelle seguenti situazioni:

- Accesso utente federato: per assegnare le autorizzazioni a una identità federata, è possibile creare un ruolo e definire le autorizzazioni per il ruolo. Quando un'identità federata viene

autenticata, l'identità viene associata al ruolo e ottiene le autorizzazioni da esso definite. Per ulteriori informazioni sulla federazione dei ruoli, consulta Creazione di un ruolo per un provider di identità di terza parte nella Guida per l'utente di IAM. Se utilizzi IAM Identity Center, configura un set di autorizzazioni. IAM Identity Center mette in correlazione il set di autorizzazioni con un ruolo in IAM per controllare a cosa possono accedere le identità dopo l'autenticazione. Per informazioni sui set di autorizzazioni, consulta Set di autorizzazioni nella Guida per l'utente di AWS IAM Identity Center .

- Autorizzazioni utente IAM temporanee: un utente IAM o un ruolo può assumere un ruolo IAM per ottenere temporaneamente autorizzazioni diverse per un'attività specifica.
- Accesso multi-account: è possibile utilizzare un ruolo IAM per permettere a un utente (un principale affidabile) con un account diverso di accedere alle risorse nell'account. I ruoli sono lo strumento principale per concedere l'accesso multi-account. Tuttavia, con alcuni Servizi AWS, è possibile allegare una policy direttamente a una risorsa (anziché utilizzare un ruolo come proxy). Per informazioni sulle differenze tra ruoli e policy basate su risorse per l'accesso multi-account, consulta Differenza tra i ruoli IAM e le policy basate su risorse nella Guida per l'utente di IAM.
- Accesso a più servizi: alcuni Servizi AWS utilizzano le funzionalità di altri Servizi AWS. Ad esempio, quando effettui una chiamata in un servizio, è comune che tale servizio esegua applicazioni in Amazon EC2 o archivi oggetti in Amazon S3. Un servizio può eseguire questa operazione utilizzando le autorizzazioni dell'entità chiamante, utilizzando un ruolo di servizio o utilizzando un ruolo collegato al servizio.
  - Sessioni di accesso diretto (FAS): quando utilizzi un utente o un ruolo IAM per eseguire azioni AWS, sei considerato un preside. Quando si utilizzano alcuni servizi, è possibile eseguire un'azione che attiva un'altra azione in un servizio diverso. FAS utilizza le autorizzazioni del principale che chiama an Servizio AWS, combinate con la richiesta Servizio AWS per effettuare richieste ai servizi downstream. Le richieste FAS vengono effettuate solo quando un servizio riceve una richiesta che richiede interazioni con altri Servizi AWS o risorse per essere completata. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le azioni. Per i dettagli delle policy relative alle richieste FAS, consulta la pagina Forward access sessions.
  - Ruolo di servizio: un ruolo di servizio è un ruolo IAM che un servizio assume per eseguire azioni per tuo conto. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione Creazione di un ruolo per delegare le autorizzazioni a un Servizio AWS nella Guida per l'utente di IAM.
  - Ruolo collegato al servizio: un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un. Servizio AWS Il servizio può assumere il ruolo per eseguire un'azione per tuo conto. I ruoli

collegati al servizio vengono visualizzati nel tuo account Account AWS e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati ai servizi, ma non modificarle.

- Applicazioni in esecuzione su Amazon EC2: puoi utilizzare un ruolo IAM per gestire le credenziali temporanee per le applicazioni in esecuzione su un'istanza EC2 e che AWS CLI effettuano richieste API. AWS Ciò è preferibile all'archiviazione delle chiavi di accesso nell'istanza EC2. Per assegnare un AWS ruolo a un'istanza EC2 e renderlo disponibile per tutte le sue applicazioni, crei un profilo di istanza collegato all'istanza. Un profilo dell'istanza contiene il ruolo e consente ai programmi in esecuzione sull'istanza EC2 di ottenere le credenziali temporanee. Per ulteriori informazioni, consulta Utilizzo di un ruolo IAM per concedere autorizzazioni ad applicazioni in esecuzione su istanze di Amazon EC2 nella Guida per l'utente di IAM.

Per informazioni sull'utilizzo dei ruoli IAM, consulta Quando creare un ruolo IAM (invece di un utente) nella Guida per l'utente di IAM.

Gestione dell'accesso con policy

Puoi controllare l'accesso AWS creando policy e collegandole a AWS identità o risorse. Una policy è un oggetto AWS che, se associato a un'identità o a una risorsa, ne definisce le autorizzazioni. AWS valuta queste politiche quando un principale (utente, utente root o sessione di ruolo) effettua una richiesta. Le autorizzazioni nelle policy determinano l'approvazione o il rifiuto della richiesta. La maggior parte delle politiche viene archiviata AWS come documenti JSON. Per ulteriori informazioni sulla struttura e sui contenuti dei documenti delle policy JSON, consulta Panoramica delle policy JSON nella Guida per l'utente di IAM.

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale principale può eseguire azioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Per concedere agli utenti l'autorizzazione a eseguire azioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM. Successivamente l'amministratore può aggiungere le policy IAM ai ruoli e gli utenti possono assumere i ruoli.

Le policy IAM definiscono le autorizzazioni relative a un'azione, a prescindere dal metodo utilizzato per eseguirla. Ad esempio, supponiamo di disporre di una policy che consente l'azione `iam:GetRole`. Un utente con tale policy può ottenere informazioni sul ruolo dall' AWS Management Console AWS CLI, dall'o dall' AWS API.

Policy basate su identità

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruoli IAM). Tali policy definiscono le azioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta Creazione di policy IAM nella Guida per l'utente di IAM.

Le policy basate su identità possono essere ulteriormente classificate come policy inline o policy gestite. Le policy inline sono integrate direttamente in un singolo utente, gruppo o ruolo. Le politiche gestite sono politiche autonome che puoi allegare a più utenti, gruppi e ruoli nel tuo Account AWS. Le politiche gestite includono politiche AWS gestite e politiche gestite dai clienti. Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta Scelta fra policy gestite e policy inline nella Guida per l'utente di IAM.

Policy basate su risorse

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Gli esempi più comuni di policy basate su risorse sono le policy di attendibilità dei ruoli IAM e le policy dei bucket Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarle per controllare l'accesso a una risorsa specifica. Quando è collegata a una risorsa, una policy definisce le azioni che un principale può eseguire su tale risorsa e a quali condizioni. È necessario specificare un principale in una policy basata sulle risorse. I principali possono includere account, utenti, ruoli, utenti federati o. Servizi AWS

Le policy basate sulle risorse sono policy inline che si trovano in tale servizio. Non puoi utilizzare le policy AWS gestite di IAM in una policy basata sulle risorse.

Liste di controllo degli accessi (ACL)

Le liste di controllo degli accessi (ACL) controllano quali principali (membri, utenti o ruoli dell'account) hanno le autorizzazioni per accedere a una risorsa. Le ACL sono simili alle policy basate su risorse, sebbene non utilizzino il formato del documento di policy JSON.

Amazon S3 e Amazon VPC sono esempi di servizi che supportano gli ACL. AWS WAF Per maggiori informazioni sulle ACL, consulta Panoramica delle liste di controllo degli accessi (ACL) nella Guida per gli sviluppatori di Amazon Simple Storage Service.

Altri tipi di policy

AWS supporta tipi di policy aggiuntivi e meno comuni. Questi tipi di policy possono impostare il numero massimo di autorizzazioni concesse dai tipi di policy più comuni.

- Limiti delle autorizzazioni: un limite delle autorizzazioni è una funzione avanzata nella quale si imposta il numero massimo di autorizzazioni che una policy basata su identità può concedere a un'entità IAM (utente o ruolo IAM). È possibile impostare un limite delle autorizzazioni per un'entità. Le autorizzazioni risultanti sono l'intersezione delle policy basate su identità dell'entità e i relativi limiti delle autorizzazioni. Le policy basate su risorse che specificano l'utente o il ruolo nel campo `Principal` sono condizionate dal limite delle autorizzazioni. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni sui limiti delle autorizzazioni, consulta Limiti delle autorizzazioni per le entità IAM nella Guida per l'utente di IAM.
- Politiche di controllo dei servizi (SCP): le SCP sono politiche JSON che specificano le autorizzazioni massime per un'organizzazione o un'unità organizzativa (OU) in. AWS Organizations AWS Organizations è un servizio per il raggruppamento e la gestione centralizzata di più Account AWS di proprietà dell'azienda. Se abiliti tutte le funzionalità in un'organizzazione, puoi applicare le policy di controllo dei servizi (SCP) a uno o tutti i tuoi account. L'SCP limita le autorizzazioni per le entità negli account dei membri, inclusa ciascuna. Utente root dell'account AWS Per ulteriori informazioni su organizzazioni e policy SCP, consulta la pagina sulle Policy di controllo dei servizi nella Guida per l'utente di AWS Organizations .
- Policy di sessione: le policy di sessione sono policy avanzate che vengono trasmesse come parametro quando si crea in modo programmatico una sessione temporanea per un ruolo o un utente federato. Le autorizzazioni della sessione risultante sono l'intersezione delle policy basate su identità del ruolo o dell'utente e le policy di sessione. Le autorizzazioni possono anche provenire da una policy basata su risorse. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni, consulta Policy di sessione nella Guida per l'utente di IAM.

Più tipi di policy

Quando più tipi di policy si applicano a una richiesta, le autorizzazioni risultanti sono più complicate da comprendere. Per sapere come si AWS determina se consentire una richiesta quando sono coinvolti più tipi di policy, consulta Logica di valutazione delle policy nella IAM User Guide.

Controllo degli accessi

Puoi avere credenziali valide per autenticare le tue richieste, ma a meno che tu non disponga delle autorizzazioni non puoi creare o accedere alle risorse Step Functions. Ad esempio, devi disporre delle autorizzazioni per invocare obiettivi AWS Lambda Amazon Simple Notification Service (Amazon SNS) e Amazon Simple Queue Service (Amazon SQS) associati alle tue regole Step Functions.

Le seguenti sezioni descrivono come gestire le autorizzazioni per Step Functions.

- Creazione di un ruolo IAM per la tua macchina a stati
- Creazione di autorizzazioni IAM granulari per utenti non amministratori
- Endpoint Amazon VPC per Step Functions
- Politiche IAM per servizi integrati
- Politiche IAM per l'utilizzo dello stato della mappa distribuita

Azioni politiche per Step Functions

Supporta le azioni di policy	Sì

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. Cioè, quale principale può eseguire azioni su quali risorse, e in quali condizioni.

L'elemento `Action` di una policy JSON descrive le azioni che è possibile utilizzare per consentire o negare l'accesso a un criterio. Le azioni politiche in genere hanno lo stesso nome dell'operazione AWS API associata. Ci sono alcune eccezioni, ad esempio le azioni di sola autorizzazione che non hanno un'operazione API corrispondente. Esistono anche alcune operazioni che richiedono più operazioni in una policy. Queste operazioni aggiuntive sono denominate operazioni dipendenti.

Includi le operazioni in una policy per concedere le autorizzazioni a eseguire l'operazione associata.

Per visualizzare un elenco delle azioni di Step Functions, vedere Resources Defined by AWS Step Functions nel Service Authorization Reference.

Le azioni politiche in Step Functions utilizzano il seguente prefisso prima dell'azione:

```
states
```

Per specificare più operazioni in una sola istruzione, occorre separarle con la virgola.

```
"Action": [
      "states:action1",
      "states:action2"
         ]
```

Per visualizzare esempi di politiche basate sull'identità di Step Functions, vedere. Esempi di policy basate sull'identità per AWS Step Functions

Risorse politiche per Step Functions

Supporta le risorse di policy	Sì

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. Cioè, quale principale può eseguire operazioni su quali risorse, e in quali condizioni.

L'elemento JSON `Resource` della policy specifica l'oggetto o gli oggetti ai quali si applica l'azione. Le istruzioni devono includere un elemento `Resource` o un elemento `NotResource`. Come best practice, specifica una risorsa utilizzando il suo nome della risorsa Amazon (ARN). Puoi eseguire questa operazione per azioni che supportano un tipo di risorsa specifico, note come autorizzazioni a livello di risorsa.

Per le azioni che non supportano le autorizzazioni a livello di risorsa, ad esempio le operazioni di elenco, utilizza un carattere jolly (*) per indicare che l'istruzione si applica a tutte le risorse.

```
"Resource": "*"
```

Per visualizzare un elenco dei tipi di risorse Step Functions e dei relativi ARN, vedere Actions Defined by AWS Step Functions nel Service Authorization Reference. Per sapere con quali azioni è possibile specificare l'ARN di ogni risorsa, vedere Risorse definite da. AWS Step Functions

Per visualizzare esempi di politiche basate sull'identità di Step Functions, vedere. Esempi di policy basate sull'identità per AWS Step Functions

Tasti relativi alle condizioni delle policy per Step Functions

Supporta le chiavi di condizione delle policy specifiche del servizio	Sì

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. Cioè, quale principale può eseguire azioni su quali risorse, e in quali condizioni.

L'elemento `Condition` (o blocco `Condition`) consente di specificare le condizioni in cui un'istruzione è in vigore. L'elemento `Condition` è facoltativo. Puoi compilare espressioni condizionali che utilizzano operatori di condizione, ad esempio uguale a o minore di, per soddisfare la condizione nella policy con i valori nella richiesta.

Se specifichi più elementi `Condition` in un'istruzione o più chiavi in un singolo elemento `Condition`, questi vengono valutati da AWS utilizzando un'operazione `AND` logica. Se si specificano più valori per una singola chiave di condizione, AWS valuta la condizione utilizzando un'operazione logica. `OR` Tutte le condizioni devono essere soddisfatte prima che le autorizzazioni dell'istruzione vengano concesse.

Puoi anche utilizzare variabili segnaposto quando specifichi le condizioni. Ad esempio, puoi autorizzare un utente IAM ad accedere a una risorsa solo se è stata taggata con il relativo nome utente IAM. Per ulteriori informazioni, consulta Elementi delle policy IAM: variabili e tag nella Guida per l'utente di IAM.

AWS supporta chiavi di condizione globali e chiavi di condizione specifiche del servizio. Per visualizzare tutte le chiavi di condizione AWS globali, consulta le chiavi di contesto delle condizioni AWS globali nella Guida per l'utente IAM.

Per visualizzare un elenco dei tasti di condizione di Step Functions, vedere Condition Keys for AWS Step Functions nel Service Authorization Reference. Per sapere con quali azioni e risorse puoi utilizzare una chiave di condizione, vedi Risorse definite da AWS Step Functions.

Per visualizzare esempi di politiche basate sull'identità di Step Functions, vedere. Esempi di policy basate sull'identità per AWS Step Functions

ACL in Step Functions

Supporta le ACL	No

Le liste di controllo degli accessi (ACL) controllano quali principali (membri, utenti o ruoli dell'account) hanno le autorizzazioni ad accedere a una risorsa. Le ACL sono simili alle policy basate su risorse, sebbene non utilizzino il formato del documento di policy JSON.

ABAC con Step Functions

Supporta ABAC (tag nelle policy)	Parziale

Il controllo dell'accesso basato su attributi (ABAC) è una strategia di autorizzazione che definisce le autorizzazioni in base agli attributi. In AWS, questi attributi sono chiamati tag. Puoi allegare tag a entità IAM (utenti o ruoli) e a molte AWS risorse. L'assegnazione di tag alle entità e alle risorse è il primo passaggio di ABAC. In seguito, vengono progettate policy ABAC per consentire operazioni quando il tag dell'entità principale corrisponde al tag sulla risorsa a cui si sta provando ad accedere.

La strategia ABAC è utile in ambienti soggetti a una rapida crescita e aiuta in situazioni in cui la gestione delle policy diventa impegnativa.

Per controllare l'accesso basato su tag, fornisci informazioni sui tag nell'elemento condizione di una policy utilizzando le chiavi di condizione `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`o `aws:TagKeys`.

Se un servizio supporta tutte e tre le chiavi di condizione per ogni tipo di risorsa, il valore per il servizio è Yes (Sì). Se un servizio supporta tutte e tre le chiavi di condizione solo per alcuni tipi di risorsa, allora il valore sarà Parziale.

Per ulteriori informazioni su ABAC, consulta Che cos'è ABAC? nella Guida per l'utente di IAM. Per visualizzare un tutorial con i passaggi per l'impostazione di ABAC, consulta Utilizzo del controllo degli accessi basato su attributi (ABAC) nella Guida per l'utente di IAM.

Utilizzo di credenziali temporanee con Step Functions

Supporta le credenziali temporanee	Sì

Alcune Servizi AWS non funzionano quando si accede utilizzando credenziali temporanee. Per ulteriori informazioni, incluse quelle che Servizi AWS funzionano con credenziali temporanee, consulta la sezione relativa alla Servizi AWS compatibilità con IAM nella IAM User Guide.

Stai utilizzando credenziali temporanee se accedi AWS Management Console utilizzando qualsiasi metodo tranne nome utente e password. Ad esempio, quando accedi AWS utilizzando il link Single Sign-On (SSO) della tua azienda, tale processo crea automaticamente credenziali temporanee. Le credenziali temporanee vengono create in automatico anche quando accedi alla console come utente e poi cambi ruolo. Per ulteriori informazioni sullo scambio dei ruoli, consulta Cambio di un ruolo (console) nella Guida per l'utente di IAM.

È possibile creare manualmente credenziali temporanee utilizzando l'API or. AWS CLI AWS È quindi possibile utilizzare tali credenziali temporanee per accedere. AWS AWS consiglia di generare dinamicamente credenziali temporanee anziché utilizzare chiavi di accesso a lungo termine. Per ulteriori informazioni, consulta Credenziali di sicurezza provvisorie in IAM.

Autorizzazioni principali per diversi servizi per Step Functions

Supporta l'inoltro delle sessioni di accesso (FAS)	Sì

Quando utilizzi un utente o un ruolo IAM per eseguire azioni AWS, sei considerato un principale. Quando si utilizzano alcuni servizi, è possibile eseguire un'azione che attiva un'altra azione in un servizio diverso. FAS utilizza le autorizzazioni del principale che chiama an Servizio AWS, in combinazione con la richiesta Servizio AWS per effettuare richieste ai servizi downstream. Le richieste FAS vengono effettuate solo quando un servizio riceve una richiesta che richiede interazioni con altri Servizi AWS o risorse per essere completata. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le azioni. Per i dettagli delle policy relative alle richieste FAS, consulta la pagina Forward access sessions.

Ruoli di servizio per Step Functions

Supporta i ruoli di servizio	Sì

Un ruolo di servizio è un ruolo IAM che un servizio assume per eseguire operazioni per tuo conto. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione Creazione di un ruolo per delegare le autorizzazioni a un Servizio AWS nella Guida per l'utente di IAM.

> **Warning**
>
> La modifica delle autorizzazioni per un ruolo di servizio potrebbe interrompere la funzionalità di Step Functions. Modifica i ruoli di servizio solo quando Step Functions fornisce indicazioni in tal senso.

Ruoli collegati ai servizi per Step Functions

Supporta i ruoli collegati ai servizi	No

Un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un. Servizio AWS Il servizio può assumere il ruolo per eseguire un'azione per tuo conto. I ruoli collegati al servizio vengono visualizzati nel tuo account Account AWS e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati ai servizi, ma non modificarle.

Per ulteriori informazioni su come creare e gestire i ruoli collegati ai servizi, consulta Servizi AWS supportati da IAM. Trova un servizio nella tabella che include un `Yes` nella colonna Service-linked role (Ruolo collegato ai servizi). Scegli il collegamento Sì per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Come AWS Step Functions funziona con IAM

Prima di utilizzare IAM per gestire l'accesso a Step Functions, scopri quali funzionalità IAM sono disponibili per l'uso con Step Functions.

Funzionalità IAM che puoi utilizzare con AWS Step Functions

Funzionalità IAM	Supporto Step Functions
Policy basate su identità	Sì
Policy basate su risorse	No
Azioni di policy	Sì
Risorse relative alle policy	Sì

Funzionalità IAM	Supporto Step Functions
Chiavi di condizione della policy (specifica del servizio)	Sì
Liste di controllo degli accessi (ACL)	No
ABAC (tag nelle policy)	Parziale
Credenziali temporanee	Sì
Autorizzazioni del principale	Sì
Ruoli di servizio	Sì
Ruoli collegati al servizio	No

Per avere una visione di alto livello di come Step Functions e altri AWS servizi funzionano con la maggior parte delle funzionalità IAM, consulta AWS i servizi che funzionano con IAM nella IAM User Guide.

Esempi di policy basate sull'identità per AWS Step Functions

Per impostazione predefinita, gli utenti e i ruoli non dispongono dell'autorizzazione per creare o modificare le risorse Step Functions. Inoltre, non possono eseguire attività utilizzando AWS Management Console, AWS Command Line Interface (AWS CLI) o AWS l'API. Per concedere agli utenti l'autorizzazione a eseguire azioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM. L'amministratore può quindi aggiungere le policy IAM ai ruoli e gli utenti possono assumere i ruoli.

Per informazioni su come creare una policy basata su identità IAM utilizzando questi documenti di policy JSON di esempio, consulta Creazione di policy IAM nella Guida per l'utente di IAM.

Per i dettagli sulle azioni e sui tipi di risorse definiti da Step Functions, incluso il formato degli ARN per ciascun tipo di risorsa, vedere Actions, Resources and Condition Keys AWS Step Functions nel Service Authorization Reference.

Argomenti

- Best practice per le policy

- Utilizzo della console Step Functions
- Consentire agli utenti di visualizzare le loro autorizzazioni

Best practice per le policy

Le politiche basate sull'identità determinano se qualcuno può creare, accedere o eliminare le risorse Step Functions nel tuo account. Queste azioni possono comportare costi aggiuntivi per l' Account AWS. Quando crei o modifichi policy basate su identità, segui queste linee guida e raccomandazioni:

- Inizia con le policy AWS gestite e passa alle autorizzazioni con privilegi minimi: per iniziare a concedere autorizzazioni a utenti e carichi di lavoro, utilizza le politiche gestite che concedono le autorizzazioni per molti casi d'uso comuni.AWS Sono disponibili nel tuo. Account AWS Ti consigliamo di ridurre ulteriormente le autorizzazioni definendo politiche gestite dai AWS clienti specifiche per i tuoi casi d'uso. Per ulteriori informazioni, consulta Policy gestite da AWS o Policy gestite da AWS per le funzionalità dei processi nella Guida per l'utente di IAM.
- Applica le autorizzazioni con privilegi minimi: quando imposti le autorizzazioni con le policy IAM, concedi solo le autorizzazioni richieste per eseguire un'attività. Puoi farlo definendo le azioni che possono essere intraprese su risorse specifiche in condizioni specifiche, note anche come autorizzazioni con privilegi minimi. Per ulteriori informazioni sull'utilizzo di IAM per applicare le autorizzazioni, consulta Policy e autorizzazioni in IAM nella Guida per l'utente di IAM.
- Condizioni d'uso nelle policy IAM per limitare ulteriormente l'accesso: per limitare l'accesso ad azioni e risorse puoi aggiungere una condizione alle tue policy. Ad esempio, è possibile scrivere una condizione di policy per specificare che tutte le richieste devono essere inviate utilizzando SSL. Puoi anche utilizzare le condizioni per concedere l'accesso alle azioni del servizio se vengono utilizzate tramite uno specifico Servizio AWS, ad esempio AWS CloudFormation. Per ulteriori informazioni, consulta la sezione Elementi delle policy JSON di IAM: condizione nella Guida per l'utente di IAM.
- Utilizzo di IAM Access Analyzer per convalidare le policy IAM e garantire autorizzazioni sicure e funzionali: IAM Access Analyzer convalida le policy nuove ed esistenti in modo che aderiscano alla sintassi della policy IAM (JSON) e alle best practice di IAM. IAM Access Analyzer offre oltre 100 controlli delle policy e consigli utili per creare policy sicure e funzionali. Per ulteriori informazioni, consulta Convalida delle policy per IAM Access Analyzer nella Guida per l'utente di IAM.
- Richiedi l'autenticazione a più fattori (MFA): se hai uno scenario che richiede utenti IAM o un utente root nel Account AWS tuo, attiva l'MFA per una maggiore sicurezza. Per richiedere la MFA quando vengono chiamate le operazioni API, aggiungi le condizioni MFA alle policy. Per ulteriori

informazioni, consulta Configurazione dell'accesso alle API protetto con MFA nella Guida per l'utente di IAM.

Per maggiori informazioni sulle best practice in IAM, consulta Best practice di sicurezza in IAM nella Guida per l'utente di IAM.

Utilizzo della console Step Functions

Per accedere alla AWS Step Functions console, è necessario disporre di un set minimo di autorizzazioni. Queste autorizzazioni devono consentirti di elencare e visualizzare i dettagli sulle risorse Step Functions presenti nel tuo Account AWS. Se crei una policy basata sull'identità più restrittiva rispetto alle autorizzazioni minime richieste, la console non funzionerà nel modo previsto per le entità (utenti o ruoli) associate a tale policy.

Non è necessario consentire autorizzazioni minime per la console per gli utenti che effettuano chiamate solo verso AWS CLI o l' AWS API. Al contrario, concedi l'accesso solo alle operazioni che corrispondono all'operazione API che stanno cercando di eseguire.

Per garantire che utenti e ruoli possano ancora utilizzare la console Step Functions, allega anche Step Functions *ConsoleAccess* o la policy *ReadOnly* AWS gestita alle entità. Per ulteriori informazioni, consulta Aggiunta di autorizzazioni a un utente nella Guida per l'utente IAM.

Consentire agli utenti di visualizzare le loro autorizzazioni

Questo esempio mostra in che modo è possibile creare una policy che consente agli utenti IAM di visualizzare le policy inline e gestite che sono cpllegate alla relativa identità utente. Questa politica include le autorizzazioni per completare questa azione sulla console o utilizzando l'API o a livello di codice. 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": "*"
        }
    ]
}
```

Politiche basate sull'identità per Step Functions

Supporta le policy basate su identità	Sì

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruolo IAM). Tali policy definiscono le azioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta Creazione di policy IAM nella Guida per l'utente di IAM.

Con le policy basate su identità di IAM, è possibile specificare quali operazioni e risorse sono consentite o respinte, nonché le condizioni in base alle quali le operazioni sono consentite o respinte. Non è possibile specificare l'entità principale in una policy basata sull'identità perché si applica all'utente o al ruolo a cui è associato. Per informazioni su tutti gli elementi utilizzabili in una policy JSON, consulta Guida di riferimento agli elementi delle policy JSON IAM nella Guida per l'utente di IAM.

Esempi di policy basate sull'identità per Step Functions

Per visualizzare esempi di politiche basate sull'identità di Step Functions, vedere. Esempi di policy basate sull'identità per AWS Step Functions

Politiche basate sulle risorse all'interno di Step Functions

Supporta le policy basate su risorse	No

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Gli esempi più comuni di policy basate su risorse sono le policy di attendibilità dei ruoli IAM e le policy dei bucket Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarle per controllare l'accesso a una risorsa specifica. Quando è collegata a una risorsa, una policy definisce le azioni che un principale può eseguire su tale risorsa e a quali condizioni. È necessario specificare un principale in una policy basata sulle risorse. I principali possono includere account, utenti, ruoli, utenti federati o. Servizi AWS

Per consentire l'accesso multi-account, puoi specificare un intero account o entità IAM in un altro account come principale in una policy basata sulle risorse. L'aggiunta di un principale multi-account a una policy basata sulle risorse rappresenta solo una parte della relazione di trust. Quando il principale e la risorsa sono diversi Account AWS, un amministratore IAM dell'account affidabile deve inoltre concedere all'entità principale (utente o ruolo) l'autorizzazione ad accedere alla risorsa. L'autorizzazione viene concessa collegando all'entità una policy basata sull'identità. Tuttavia, se una policy basata su risorse concede l'accesso a un principale nello stesso account, non sono richieste ulteriori policy basate su identità. Per ulteriori informazioni, consulta Differenza tra i ruoli IAM e le policy basate su risorse nella Guida per l'utente di IAM.

AWS politiche gestite per AWS Step Functions

Una politica AWS gestita è una politica autonoma creata e amministrata da AWS. AWS le politiche gestite sono progettate per fornire autorizzazioni per molti casi d'uso comuni, in modo da poter iniziare ad assegnare autorizzazioni a utenti, gruppi e ruoli.

Tieni presente che le policy AWS gestite potrebbero non concedere le autorizzazioni con il privilegio minimo per i tuoi casi d'uso specifici, poiché sono disponibili per tutti i clienti. AWS Consigliamo

pertanto di ridurre ulteriormente le autorizzazioni definendo policy gestite dal cliente specifiche per i tuoi casi d'uso.

Non è possibile modificare le autorizzazioni definite nelle politiche gestite. AWS Se AWS aggiorna le autorizzazioni definite in una politica AWS gestita, l'aggiornamento ha effetto su tutte le identità principali (utenti, gruppi e ruoli) a cui è associata la politica. AWS è più probabile che aggiorni una policy AWS gestita quando ne Servizio AWS viene lanciata una nuova o quando diventano disponibili nuove operazioni API per i servizi esistenti.

Per ulteriori informazioni, consultare Policy gestite da AWSnella Guida per l'utente di IAM.

AWS politica gestita: AWSStepFunctionsConsoleFullAccess

È possibile allegare la policy `AWSStepFunctionsConsoleFullAccess`alle identità IAM.

Questa politica concede le autorizzazioni di *`amministratore`* che consentono a un utente di accedere all'utilizzo della console Step Functions. Per un'esperienza di console completa, un utente può anche aver bisogno di iam: PassRole autorizzazione per altri ruoli IAM che può essere assunta dal servizio.

AWS politica gestita: AWSStepFunctionsReadOnlyAccess

È possibile allegare la policy `AWSStepFunctionsReadOnlyAccess`alle identità IAM.

Questa politica concede autorizzazioni di *`sola lettura`* che consentono a un utente o a un ruolo di elencare e descrivere macchine a stati, attività, esecuzioni, attività MapRuns, tag e alias e versioni delle macchine a stati.

AWS politica gestita: AWSStepFunctionsFullAccess

È possibile allegare la policy `AWSStepFunctionsFullAccess`alle identità IAM.

Questa politica concede le autorizzazioni *`complete`* a un utente o un ruolo per utilizzare l'API Step Functions. Per un accesso completo, un utente deve disporre dell'PassRoleautorizzazione *`iam:`* per almeno un ruolo IAM che può essere assunto dal servizio.

Step Functions aggiorna le policy AWS gestite

Visualizza i dettagli sugli aggiornamenti delle politiche AWS gestite per Step Functions da quando questo servizio ha iniziato a tenere traccia di queste modifiche. Per ricevere avvisi automatici

sulle modifiche a questa pagina, iscriviti al feed RSS nella pagina Step FunctionsCronologia dei documenti.

Modifica	Descrizione	Data
AWSStepFunctionsReadOnlyAccess: aggiornamento a una policy esistente	Step Functions ha aggiunto nuove autorizzazioni per consentire l'elenco e la lettura dei dati relativi a: tag (ListTagsForResource), mappa distribuita (ListMapRuns, DescribeMapRun), versioni e alias (DescribeStateMachineAlias, ListStateMachineAliases, ListStateMachineVersions).	02 aprile 2024
Step Functions ha iniziato a tenere traccia delle modifiche	Step Functions ha iniziato a tenere traccia delle modifiche per le sue politiche AWS gestite.	02 aprile 2024

Creazione di un ruolo IAM per la tua macchina a stati

AWS Step Functions può eseguire codice e accedere alle AWS risorse (ad esempio invocando una AWS Lambda funzione). Per garantire la sicurezza, è necessario concedere l'accesso a Step Functions a quelle risorse utilizzando un ruolo IAM.

I Tutorial per Step Functions contenuti di questa guida consentono di sfruttare i ruoli IAM generati automaticamente e validi per la AWS regione in cui viene creata la macchina a stati. Tuttavia, puoi creare il tuo ruolo IAM per una macchina a stati.

Quando si crea una policy IAM da utilizzare per le macchine a stati, la policy deve includere le autorizzazioni che si desidera vengano concesse alle macchine a stati. Puoi utilizzare una policy AWS gestita esistente come esempio oppure puoi creare una policy personalizzata partendo da zero che soddisfi le tue esigenze specifiche. Per ulteriori informazioni, consulta Creazione di politiche IAM nella Guida per l'utente IAM

Per creare il tuo ruolo IAM per una macchina a stati, segui i passaggi in questa sezione.

In questo esempio, crei un ruolo IAM con il permesso di richiamare una funzione Lambda.

Creare un ruolo per Step Functions

1. Accedi alla console IAM, quindi scegli Ruoli, Crea ruolo.
2. Nella pagina Seleziona entità affidabile, in AWS servizio, seleziona Step Functions dall'elenco, quindi scegli Avanti: Autorizzazioni.
3. Nella pagina Attached permissions policy (Policy autorizzazioni collegate) scegliere Next: Review (Successivo: Esamina).
4. Nella pagina Review (Esamina), immettere `StepFunctionsLambdaRole` in Role Name (Nome ruolo), quindi scegliere Create role (Crea ruolo).

   Il ruolo IAM viene visualizzato nell'elenco dei ruoli.

Per ulteriori informazioni sulle autorizzazioni e le policy IAM, consulta Access Management nella IAM User Guide.

Evita problemi secondari confusi tra diversi servizi

Con "confused deputy" si intende un problema di sicurezza in cui un'entità che non dispone dell'autorizzazione per eseguire una certa operazione può costringere un'entità con più privilegi a eseguire tale operazione. Nel AWS, l'impersonificazione tra servizi può causare il problema del sostituto confuso. La rappresentazione tra servizi può verificarsi quando un servizio (il servizio chiamante) effettua una chiamata a un altro servizio (il servizio chiamato). Questo tipo di impersonificazione può avvenire tra diversi account e tra diversi servizi. Il servizio chiamante può essere manipolato per utilizzare le proprie autorizzazioni e agire sulle risorse di un altro cliente, a cui normalmente non avrebbe accesso.

Per evitare la confusione dei deputati, AWS fornisce strumenti che consentono di proteggere i dati relativi a tutti i servizi, con responsabili del servizio a cui è stato consentito l'accesso alle risorse del vostro account. Questa sezione si concentra sulla prevenzione interservizio dei deputati confusi, specifica per AWS Step Functions; tuttavia, puoi saperne di più su questo argomento nella sezione dedicata ai problemi dei deputati confusi della IAM User Guide.

Ti consigliamo di utilizzare le chiavi di contesto `aws:SourceArn` e `aws:SourceAccount` global condition nelle politiche delle risorse per limitare le autorizzazioni che Step Functions concede a un altro servizio di accedere alle tue risorse. Utilizza `aws:SourceArn` se desideri consentire

l'associazione di una sola risorsa all'accesso tra servizi. Utilizza `aws:SourceAccount` se desideri consentire l'associazione di qualsiasi risorsa in tale account all'uso tra servizi.

Il modo più efficace per proteggersi dal problema "confused deputy" è quello di usare la chiave di contesto della condizione globale `aws:SourceArn`con l'ARN completo della risorsa. Se non conosci l'ARN completo della risorsa o se stai specificando più risorse, usa la chiave di condizione di contesto `aws:SourceArn` globale con caratteri jolly (*) per le parti sconosciute dell'ARN. Ad esempio, `arn:aws:states:*:111122223333:*`.

Ecco un esempio di policy affidabile che mostra come utilizzare `aws:SourceArn` e `aws:SourceAccount` con Step Functions per prevenire la confusa questione del vice.

```
{
  "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"
           }
        }
     }
  ]
}
```

Collegamento di una policy inline

Step Functions può controllare altri servizi direttamente in uno `Task` stato. Allega policy in linea per consentire a Step Functions di accedere alle azioni API dei servizi che devi controllare.

1. Apri la console IAM, scegli Roles, cerca il tuo ruolo Step Functions e seleziona quel ruolo.
2. Selezionare Add inline policy (Aggiungi policy inline).

3. Utilizzare il Visual editor (Editor visivo) o la scheda JSON per creare policy per il ruolo.

Per ulteriori informazioni su come AWS Step Functions controllare altri AWS servizi, consultaUtilizzo AWS Step Functions con altri servizi.

> **Note**
>
> Per esempi di policy IAM create dalla console Step Functions, vediPolitiche IAM per servizi integrati.

Creazione di autorizzazioni IAM granulari per utenti non amministratori

Le policy gestite predefinite in IAM, ad esempio`ReadOnly`, non coprono completamente tutti i tipi di autorizzazioni. AWS Step Functions Questa sezione descrive i diversi tipi di autorizzazione e presenta alcune configurazioni di esempio.

Step Functions dispone di quattro categorie di autorizzazioni. In base al tipo di accesso che desideri offrire a un utente, puoi controllare l'accesso utilizzando le autorizzazioni in queste categorie.

Autorizzazioni a livello di servizio

Da applicare ai componenti dell'API che non agiscono su una risorsa specifica.

Autorizzazioni a livello di macchina a stati

Da applicare a tutti i componenti di un'API che agiscono su una macchina a stati specifica.

Autorizzazioni a livello di esecuzione

Da applicare a tutti i componenti di un'API che agiscono su un'esecuzione specifica.

Autorizzazioni a livello di attività

Da applicare a tutti i componenti di un'API che agiscono su un'attività specifica o su una determinata istanza di un'attività.

Autorizzazioni a livello di servizio

Questo livello di autorizzazione si applica a tutte le operazioni API che non agiscono su una risorsa specifica. Tra queste vi sono `CreateStateMachine`, `CreateActivity`, `ListStateMachines` e `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"
      ]
    }
  ]
}
```

Autorizzazioni a livello di macchina a stati

Questo livello di autorizzazione si applica a tutte le azioni API che agiscono su una macchina a stati specifica. Queste operazioni API richiedono l'Amazon Resource Name (ARN) della macchina a stati come parte della richiesta, ad esempio `DeleteStateMachine` `DescribeStateMachineStartExecution`, e. `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*"
      ]
    }
  ]
}
```

Autorizzazioni a livello di esecuzione

Questo livello di autorizzazione si applica a tutte le azioni API che agiscono su un'esecuzione specifica. Queste operazioni API necessitano l'ARN dell'esecuzione come parte della richiesta, ad esempio `DescribeExecution`, `GetExecutionHistory` e `StopExecution`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeExecution",
        "states:DescribeStateMachineForExecution",
        "states:GetExecutionHistory",
        "states:StopExecution"
      ],
      "Resource": [
        "arn:aws:states:*:*:execution:*:ExecutionPrefix*"
      ]
    }
  ]
}
```

Autorizzazioni a livello di attività

Questo livello di autorizzazione si applica a tutte le azioni API che agiscono su un'attività specifica o su una sua determinata istanza. Queste operazioni API richiedono l'ARN

dell'attività o il token dell'istanza come parte della richiesta, ad esempio DeleteActivity DescribeActivityGetActivityTask, e. SendTaskHeartbeat

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeActivity",
        "states:DeleteActivity",
        "states:GetActivityTask",
        "states:SendTaskHeartbeat"
      ],
      "Resource": [
        "arn:aws:states:*:*:activity:ActivityPrefix*"
      ]
    }
  ]
}
```

Accesso a risorse Account AWS in altri flussi di lavoro

Step Functions fornisce l'accesso tra account a risorse configurate Account AWS in diversi flussi di lavoro. Utilizzando le integrazioni dei servizi Step Functions, puoi richiamare qualsiasi AWS risorsa tra account anche se questa Servizio AWS non supporta policy basate sulle risorse o chiamate tra account.

Ad esempio, supponiamo di possederne due, denominate Development and Account AWS Testing, nella stessa unità. Regione AWS Utilizzando l'accesso tra account, il flusso di lavoro nell'account Development può accedere a risorse, come bucket Amazon S3, tabelle Amazon DynamoDB e funzioni Lambda disponibili nell'account Testing.

> **Important**
>
> I ruoli IAM e le policy basate sulle risorse delegano l'accesso tra account solo all'interno di una singola partizione. Ad esempio, si supponga di disporre di un account nella regione Stati Uniti occidentali (California settentrionale) nella partizione `aws` standard. Hai anche un account nella regione Cina (Pechino) nella partizione `aws-cn`. Non è possibile utilizzare

una policy basata sulle risorse Amazon S3 nel tuo account nella regione Cina (Pechino) per consentire l'accesso agli utenti del tuo account `aws` standard.

Per ulteriori informazioni sull'accesso tra account, consulta la logica di valutazione delle politiche tra account nella IAM User Guide.

Sebbene ognuno Account AWS mantenga il controllo completo sulle proprie risorse, con Step Functions puoi riorganizzare, scambiare, aggiungere o rimuovere passaggi nei tuoi flussi di lavoro senza la necessità di personalizzare alcun codice. Puoi farlo anche se i processi cambiano o le applicazioni si evolvono.

Puoi anche richiamare l'esecuzione di macchine a stati annidate in modo che siano disponibili su diversi account. In questo modo i flussi di lavoro vengono separati e isolati in modo efficiente. Quando utilizzi il modello di integrazione dei `.sync`servizi nei flussi di lavoro che accedono a un altro flusso di lavoro Step Functions in un altro account, Step Functions utilizza il polling che utilizza la quota assegnata. Per ulteriori informazioni, consulta Esegui un processo (.sync).

> **Note**
>
> Attualmente, l'integrazione AWS SDK tra regioni e l'accesso alle AWS risorse tra regioni non sono disponibili in Step Functions.

Indice

- Concetti chiave in questo argomento
- Richiamo di risorse tra account diversi
- Tutorial: Accesso alle risorse di più account AWS
- Accesso su più account per il modello di integrazione.sync

Concetti chiave in questo argomento

Ruolo di esecuzione

Un ruolo IAM utilizzato da Step Functions per eseguire codice e accedere a AWS risorse, come l'azione Invoke della AWS Lambda funzione.

Integrazione dei servizi

Le azioni dell'API di integrazione AWS SDK che possono essere richiamate dall'interno di uno `Task` stato nei flussi di lavoro.

account di origine

Un Account AWS che possiede la macchina a stati e ne ha avviato l'esecuzione.

account di destinazione

E Account AWS a cui si effettuano chiamate tra più account.

ruolo bersaglio

Un ruolo IAM nell'account di destinazione che la macchina a stati assume per effettuare chiamate alle risorse di proprietà dell'account di destinazione.

Esegui un Job (.sync)

Un modello di integrazione dei servizi utilizzato per chiamare servizi, ad esempio AWS Batch. Inoltre, fa sì che una macchina a stati Step Functions attenda il completamento di un lavoro prima di passare allo stato successivo. Per indicare che Step Functions dovrebbe attendere, aggiungi il `.sync` suffisso nel `Resource` campo della definizione `Task` dello stato.

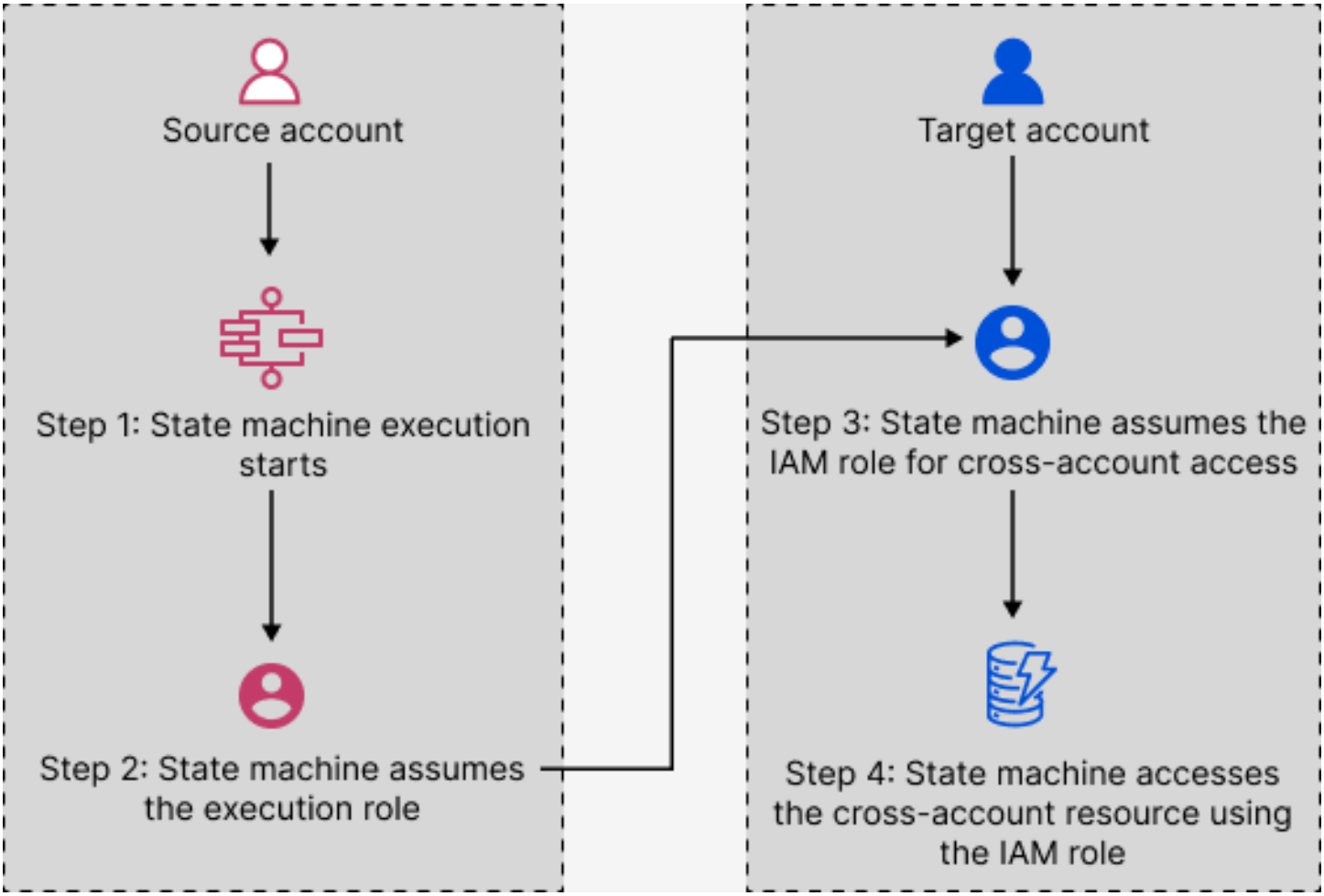
Richiamo di risorse tra account diversi

Per richiamare una risorsa che coinvolge più account nei flussi di lavoro, procedi come segue:

1. Crea un ruolo IAM nell'account di destinazione che contiene la risorsa. Questo ruolo concede all'account di origine, contenente la macchina a stati, le autorizzazioni per accedere alle risorse dell'account di destinazione.
2. Nella definizione dello `Task` stato, specifica il ruolo IAM di destinazione che deve essere assunto dalla macchina a stati prima di richiamare la risorsa tra account.
3. Modifica la policy di fiducia nel ruolo IAM di destinazione per consentire all'account di origine di assumere temporaneamente questo ruolo. La policy di fiducia deve includere l'Amazon Resource Name (ARN) della macchina a stati definita nell'account di origine. Inoltre, definisci le autorizzazioni appropriate nel ruolo IAM di destinazione per chiamare la AWS risorsa.
4. Aggiorna il ruolo di esecuzione dell'account di origine per includere l'autorizzazione richiesta per assumere il ruolo IAM di destinazione.

Per vedere un esempio, consulta Tutorial: Accesso alle risorse di più account AWS.

> **Note**
>
> Puoi configurare la tua macchina a stati in modo che assuma un ruolo IAM per l'accesso alle risorse da più Account AWS risorse. Tuttavia, una macchina a stati può assumere un solo ruolo IAM alla volta.



Tutorial: Accesso alle risorse di più account AWS

Con il supporto per l'accesso tra account in Step Functions, puoi condividere risorse configurate in diversi modi Account AWS. In questo tutorial, ti illustreremo il processo di accesso a una funzione Lambda multiaccount definita in un account chiamato Production. Questa funzione viene richiamata da una macchina a stati in un account chiamato Development. In questo tutorial, l'account

Development viene definito account di origine e l'account di produzione è l'account di destinazione contenente il ruolo IAM di destinazione.

Per iniziare, nella definizione del tuo `Task` stato, specifichi il ruolo IAM di destinazione che la macchina a stati deve assumere prima di richiamare la funzione Lambda tra account. Quindi, modifica la policy di fiducia nel ruolo IAM di destinazione per consentire all'account di origine di assumere temporaneamente il ruolo di destinazione. Inoltre, per chiamare la AWS risorsa, definisci le autorizzazioni appropriate nel ruolo IAM di destinazione. Infine, aggiorna il ruolo di esecuzione dell'account di origine per specificare l'autorizzazione richiesta per assumere il ruolo di destinazione.

Puoi configurare la tua macchina a stati in modo che assuma un ruolo IAM per l'accesso alle risorse da più fonti Account AWS. Tuttavia, una macchina a stati può assumere un solo ruolo IAM alla volta in base alla definizione `Task` dello stato.

> **Note**
>
> Attualmente, l'integrazione AWS SDK tra regioni e l'accesso alle AWS risorse tra regioni non sono disponibili in Step Functions.

Indice

- Prerequisiti
- Passaggio 1: aggiorna la definizione dello stato dell'attività per specificare il ruolo di destinazione
- Passaggio 2: aggiorna la politica di fiducia del ruolo di destinazione
- Passaggio 3: aggiungi l'autorizzazione richiesta nel ruolo di destinazione
- Passaggio 4: Aggiungere l'autorizzazione nel ruolo di esecuzione per assumere il ruolo di destinazione

Prerequisiti

- Questo tutorial utilizza l'esempio di una funzione Lambda per dimostrare come configurare l'accesso tra account. Puoi usare qualsiasi altra AWS risorsa, ma assicurati di aver configurato la risorsa in un account diverso.

> **Important**
>
> I ruoli IAM e le policy basate sulle risorse delegano l'accesso tra account solo all'interno di una singola partizione. Ad esempio, si supponga di disporre di un account nella regione Stati Uniti occidentali (California settentrionale) nella partizione `aws` standard. Hai anche un account nella regione Cina (Pechino) nella partizione `aws-cn`. Non è possibile utilizzare una policy basata sulle risorse Amazon S3 nel tuo account nella regione Cina (Pechino) per consentire l'accesso agli utenti del tuo account `aws` standard.

- Prendi nota dell'Amazon Resource Name (ARN) della risorsa per più account in un file di testo. Più avanti in questo tutorial, fornirai questo ARN nella definizione dello `Task` stato della tua macchina a stati. Di seguito è riportato un esempio di ARN della funzione Lambda:

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

- Assicurati di aver creato il ruolo IAM di destinazione che la macchina a stati deve assumere.

Passaggio 1: aggiorna la definizione dello stato dell'attività per specificare il ruolo di destinazione

`Task`Nello stato del flusso di lavoro, aggiungi un `Credentials` campo contenente l'identità che la macchina a stati deve assumere prima di richiamare la funzione Lambda tra account.

La procedura seguente mostra come accedere a una funzione Lambda tra account denominata. `Echo` Puoi chiamare qualsiasi AWS risorsa seguendo questi passaggi.

1. Apri la console Step Functions e scegli Crea macchina a stati.
2. Nella pagina Scegli il metodo di creazione, scegli Progetta visivamente il tuo flusso di lavoro e mantieni tutte le selezioni predefinite.
3. Per aprire Workflow Studio, scegli Avanti.
4. Nella scheda Azioni, trascina uno `Task` stato sull'area di disegno. Questo richiama la funzione Lambda tra account diversi che utilizza questo stato. `Task`
5. Nella scheda Configurazione, procedi come segue:

    a. Rinomina lo stato in. **`Cross-account call`**

    b. Per Nome funzione, scegliete Inserisci il nome della funzione, quindi immettete l'ARN della funzione Lambda nella casella. Ad esempio, `arn:aws:lambda:us-east-2:111122223333:function:Echo`.

c. Per Provide IAM role ARN, specifica l'ARN del ruolo IAM di destinazione. Ad esempio, `arn:aws:iam::111122223333:role/LambdaRole`.

> **Tip**
>
> In alternativa, puoi anche specificare un percorso di riferimento a una coppia chiave-valore esistente nell'input JSON dello stato che contiene l'ARN del ruolo IAM. Per fare ciò, scegli Ottieni il ruolo IAM ARN in fase di esecuzione dall'input dello stato. Per un esempio di specificazione di un valore utilizzando un percorso di riferimento, consulta. Specificare JSONPath come ARN del ruolo IAM

6. Seleziona Successivo.
7. Nella pagina Rivedi codice generato, scegli Avanti.
8. Nella pagina Specificare le impostazioni della macchina a stati, specificate i dettagli per la nuova macchina a stati, come il nome, le autorizzazioni e il livello di registrazione.
9. Scegli Create state machine (Crea macchina a stati).
10. Prendi nota del ruolo IAM ARN della macchina a stati e dell'ARN della macchina a stati in un file di testo. Dovrai fornire questi ARN nella policy di fiducia dell'account di destinazione.

La definizione `Task` dello stato dovrebbe ora essere simile alla definizione seguente.

```
{
  "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
    }
  }
}
```

Passaggio 2: aggiorna la politica di fiducia del ruolo di destinazione

Il ruolo IAM deve esistere nell'account di destinazione ed è necessario modificarne la politica di attendibilità per consentire all'account di origine di assumere temporaneamente questo ruolo. Inoltre, puoi controllare chi può assumere il ruolo IAM di destinazione.

Dopo aver creato la relazione di fiducia, un utente dell'account di origine può utilizzare l'operazione AssumeRoleAPI AWS Security Token Service (AWS STS). Questa operazione fornisce credenziali di sicurezza temporanee che consentono l'accesso alle AWS risorse in un account di destinazione.

1. Aprire la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel riquadro di navigazione della console, scegli Ruoli, quindi utilizza la casella di ricerca per cercare il ruolo IAM di destinazione. Ad esempio, *LambdaRole*.
3. Seleziona la scheda Relazioni di attendibilità
4. Scegli Modifica politica di fiducia e incolla la seguente politica di fiducia. Assicurati di sostituire il Account AWS numero e l'ARN del ruolo IAM. Il `sts:ExternalId` campo controlla ulteriormente chi può assumere il ruolo. Il nome della macchina a stati deve includere solo i caratteri supportati dall' AWS Security Token Service `AssumeRole`API. Per ulteriori informazioni, consulta la AssumeRolesezione AWS Security Token Service API Reference.

```
{
  "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. Tieni aperta questa finestra e procedi al passaggio successivo per ulteriori azioni.

Passaggio 3: aggiungi l'autorizzazione richiesta nel ruolo di destinazione

Le autorizzazioni nelle policy IAM determinano se una richiesta specifica è consentita o negata. Il ruolo IAM di destinazione deve disporre dell'autorizzazione corretta per richiamare la funzione Lambda.

1. Scegli la scheda Autorizzazioni.
2. Scegli Aggiungi autorizzazioni, quindi seleziona Crea policy inline.
3. Scegli la scheda JSON e sostituisci il contenuto esistente con la seguente autorizzazione. Assicurati di sostituire l'ARN della funzione Lambda.

```
{
  "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. Scegli Verifica policy.
5. Nella pagina Rivedi politica, inserisci un nome per l'autorizzazione, quindi scegli Crea politica.

Passaggio 4: Aggiungere l'autorizzazione nel ruolo di esecuzione per assumere il ruolo di destinazione

Step Functions non genera automaticamente la AssumeRolepolicy per tutte le integrazioni di servizi tra account. È necessario aggiungere l'autorizzazione richiesta nel ruolo di esecuzione della macchina a stati per consentirle di assumere un ruolo IAM di destinazione in uno o più. Account AWS

1. Apri il ruolo di esecuzione della tua macchina a stati nella console IAM all'indirizzo https://console.aws.amazon.com/iam/. Per farlo:

a. Apri la macchina a stati che hai creato nel passaggio 1 nell'account di origine.

b. Nella pagina dei dettagli della macchina a stati, scegli il ruolo IAM ARN.

2. Nella scheda Autorizzazioni, scegli Aggiungi autorizzazioni, quindi scegli Crea policy in linea.

3. Scegli la scheda JSON e sostituisci il contenuto esistente con la seguente autorizzazione. Assicurati di sostituire l'ARN della funzione Lambda.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::111122223333:role/LambdaRole"  // The target role
 to be assumed
    }
  ]
}
```

4. Scegli Verifica policy.

5. Nella pagina Rivedi politica, inserisci un nome per l'autorizzazione, quindi scegli Crea politica.

Accesso su più account per il modello di integrazione.sync

Quando si utilizzano i modelli di integrazione dei `.sync` servizi nei flussi di lavoro, Step Functions esegue il polling della risorsa interaccount richiamata per confermare che l'attività è stata completata. Ciò causa un leggero ritardo tra il momento effettivo di completamento dell'attività e il momento in cui Step Functions riconosce l'attività come completa. Il ruolo IAM di destinazione necessita delle autorizzazioni necessarie per una `.sync` chiamata per completare questo ciclo di polling. A tale scopo, il ruolo IAM di destinazione deve disporre di una policy di fiducia che consenta all'account di origine di assumerlo. Inoltre, il ruolo IAM di destinazione necessita delle autorizzazioni necessarie per completare il ciclo di polling.

> **Note**
>
> Per i flussi di lavoro Express annidati, `arn:aws:states:::states:startExecution.sync` non è attualmente supportato. Usare invece `arn:aws:states:::aws-sdk:sfn:startSyncExecution`.

Aggiornamento della politica di fiducia per le chiamate.sync

Aggiorna la policy di fiducia del tuo ruolo IAM di destinazione come mostrato nell'esempio seguente. Il `sts:ExternalId` campo controlla ulteriormente chi può assumere il ruolo. Il nome della macchina a stati deve includere solo i caratteri supportati dall' AWS Security Token Service `AssumeRole`API. Per ulteriori informazioni, consulta la AssumeRolesezione AWS Security Token Service API Reference.

```
{
  "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"
        }
      }
    }
  ]
}
```

Autorizzazioni richieste per le chiamate.sync

Per concedere le autorizzazioni richieste per la tua macchina a stati, aggiorna le autorizzazioni richieste per il ruolo IAM di destinazione. Per ulteriori informazioni, consulta the section called “Politiche IAM per servizi integrati”. Le EventBridge autorizzazioni Amazon delle politiche di esempio non sono richieste. Ad esempio, per avviare una macchina a stati, aggiungi le seguenti autorizzazioni.

```
{
  "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:*"
      ]
    }
  ]
}
```

Endpoint Amazon VPC per Step Functions

Se utilizzi Amazon Virtual Private Cloud (Amazon VPC) per ospitare AWS le tue risorse, puoi stabilire una connessione tra Amazon VPC e i flussi di lavoro. AWS Step Functions Puoi utilizzare questa connessione con i flussi di lavoro di Step Functions senza dover accedere alla rete Internet pubblica. Gli endpoint Amazon VPC sono supportati dai flussi di lavoro Standard, dai flussi di lavoro Express e dai flussi di lavoro Synchronous Express.

Amazon VPC ti consente di avviare AWS risorse in una rete virtuale personalizzata. Puoi utilizzare un VPC per controllare le impostazioni di rete, come l'intervallo di indirizzi IP, le sottoreti, le tabelle di routing e i gateway di rete. Per ulteriori informazioni sui VPC, consulta la Guida per l'utente di Amazon VPC.

Per connettere Amazon VPC a Step Functions, devi prima definire un endpoint VPC di interfaccia, che ti permetta di connettere il tuo VPC ad altri servizi. AWS L'endpoint offre una connettività scalabile e affidabile senza necessità di disporre di un gateway Internet, un'istanza NAT (Network Address Translation) o una connessione VPN. Per ulteriori informazioni, consulta Endpoint VPC di interfaccia (AWS PrivateLink) nella Guida per l'utente di Amazon VPC.

Creazione dell'endpoint

Puoi creare un AWS Step Functions endpoint nel tuo VPC utilizzando, AWS Management Console AWS Command Line Interface the AWS CLI(), AWS un SDK, AWS Step Functions l'API o. AWS CloudFormation

Per informazioni sulla creazione e sulla configurazione di un endpoint utilizzando la console Amazon VPC o il AWS CLI, consulta la sezione Creazione di un endpoint di interfaccia nella Guida per l'utente di Amazon VPC.

> **Note**
>
> Quando crei un endpoint, specifica Step Functions come servizio a cui desideri connettere il tuo VPC. Nella console Amazon VPC, i nomi dei servizi variano in base alla AWS regione. Ad esempio, se scegli US East (Virginia settentrionale), il nome del servizio per Standard Workflows ed Express Workflows è com.amazonaws.us-east-1.states e il nome di servizio per Synchronous Express Workflows è com.amazonaws.us-east-1.sync-states.

> **Note**
>
> È possibile utilizzare gli endpoint VPC senza sovrascrivere l'endpoint nell'SDK tramite DNS privato. Tuttavia, se desideri sovrascrivere l'endpoint nell'SDK for Synchronous Express Workflows, devi impostare la configurazione su. `DisableHostPrefixInjection` `true` Esempio (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();
> ```

Per informazioni sulla creazione e configurazione di un endpoint utilizzando AWS CloudFormation, consulta la risorsa AWS: :EC2: :VPCEndpoint nella Guida per l'utente.AWS CloudFormation

Politiche degli endpoint Amazon VPC

Per controllare l'accesso alla connettività a Step Functions, puoi allegare una policy per gli endpoint AWS Identity and Access Management (IAM) durante la creazione di un endpoint Amazon VPC.

Puoi creare regole IAM complesse collegando più policy per gli endpoint. Per ulteriori informazioni, consultare:

- Policy degli endpoint di Amazon Virtual Private Cloud per Step Functions
- Creazione di autorizzazioni IAM granulari per utenti non amministratori
- Controllo dell'accesso ai servizi con endpoint VPC

Policy degli endpoint di Amazon Virtual Private Cloud per Step Functions

Puoi creare una policy di endpoint Amazon VPC per Step Functions in cui specifichi quanto segue:

- Il principale che può eseguire azioni.
- Le azioni che possono essere eseguite.
- Le risorse in cui è possibile eseguire le operazioni.

L'esempio seguente mostra una policy per gli endpoint di Amazon VPC che consente a un utente di creare macchine a stati e nega a tutti gli altri utenti l'autorizzazione a eliminare le macchine a stati. La policy di esempio concede inoltre l'autorizzazione di esecuzione a tutti gli utenti .

```
{
    "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": "*"
        }
    ]
}
```

Per ulteriori informazioni sulla creazione delle policy di endpoint, consulta:

- Creazione di autorizzazioni IAM granulari per utenti non amministratori
- Controllo dell'accesso ai servizi con endpoint VPC

Politiche IAM per servizi integrati

Quando si crea una macchina a stati nella AWS Step Functions console, Step Functions produce una policy AWS Identity and Access Management (IAM) basata sulle risorse utilizzate nella definizione della macchina a stati come segue:

- Se la tua macchina a stati utilizza una delle integrazioni ottimizzate, Step Functions creerà una policy con le autorizzazioni e i ruoli necessari per la tua macchina a stati. (Eccezione: MediaConvert l'integrazione richiede la configurazione manuale delle autorizzazioni, vedi.) Politiche IAM per AWS Elemental MediaConvert
- Se la tua macchina a stati utilizza una delle integrazioni AWS SDK, verrà creato un ruolo IAM con autorizzazioni parziali. Successivamente, puoi utilizzare la console IAM per aggiungere eventuali policy relative ai ruoli mancanti.

Gli esempi seguenti mostrano come Step Functions genera una policy IAM basata sulla definizione della macchina a stati. Le voci nel codice di esempio, come *`[[resourceName]]`*, vengono sostituite con le risorse statiche elencate nella definizione della tua macchina a stati. Se disponi di più risorse statiche, ci sarà una voce per ognuna nel ruolo IAM.

Risorse dinamiche e statiche

Le risorse statiche sono definite direttamente nello stato task della tua macchina a stati. Quando includi le informazioni sulle risorse che desideri chiamare direttamente negli stati delle attività, Step Functions crea un ruolo IAM solo per quelle risorse.

Le risorse dinamiche sono quelle che vengono trasferite all'input dello stato e sono accessibili attraverso un percorso (consulta Percorsi). Se stai passando risorse dinamiche alla tua attività, Step Functions creerà una politica più privilegiata che specifica:. `"Resource": "*"`

Autorizzazioni aggiuntive per le attività che utilizzano il pattern Run a Job

Per le attività che utilizzano il pattern Run a Job (quelle che terminano con `.sync`), sono necessarie autorizzazioni aggiuntive per monitorare e ricevere una risposta dalle azioni API dei servizi connessi. Le politiche correlate includono più autorizzazioni rispetto alle attività che utilizzano i modelli Request Response o Wait for Callback. Modelli di integrazione dei serviziPer informazioni sulle attività sincrone, vedere.

> **Note**
>
> È necessario fornire autorizzazioni aggiuntive per le integrazioni di servizi che supportano il pattern Run a Job (.sync).

Step Functions utilizza due metodi per monitorare lo stato di un lavoro quando un lavoro viene eseguito su un servizio connesso, sondaggi ed eventi.

Il polling richiede l'autorizzazione `Describe` o le azioni `Get` dell'API, come `ecs:DescribeTasks` o. `glue:GetJobRun` Se nel tuo ruolo mancano queste autorizzazioni, Step Functions potrebbe non essere in grado di determinare lo stato del tuo lavoro. Questo perché alcune integrazioni di servizi Run a Job (.sync) non supportano EventBridge gli eventi e alcuni servizi inviano eventi solo se possibile.

Gli eventi inviati dai AWS servizi ad Amazon EventBridge vengono indirizzati a Step Functions utilizzando una regola gestita e richiedono autorizzazioni per `events:PutTargets` `events:PutRule`, `eevents:DescribeRule`. Se nel tuo ruolo mancano queste autorizzazioni, potrebbe verificarsi un ritardo prima che Step Functions venga a conoscenza del completamento del tuo lavoro. Per ulteriori informazioni sugli EventBridge eventi, consulta Events from AWS services.

> **Note**
>
> Per le attività Run a Job (.sync) che supportano sia il polling che gli eventi, l'attività potrebbe comunque essere completata correttamente utilizzando gli eventi. Ciò può verificarsi anche se al ruolo non sono disponibili le autorizzazioni necessarie per il polling. In questo caso, potresti non accorgerti immediatamente che i permessi di sondaggio sono errati o mancanti. Nel raro caso in cui l'evento non venga inviato o elaborato da Step Functions, l'esecuzione potrebbe bloccarsi. Per verificare che i permessi di polling siano configurati correttamente, è possibile eseguire un'esecuzione in un ambiente senza EventBridge eventi nei seguenti modi:

- Elimina la regola gestita da EventBridge, che è responsabile dell'inoltro degli eventi a Step Functions. Questa regola gestita è condivisa da tutte le macchine a stati del tuo account, quindi dovresti eseguire questa azione solo in un account di test o di sviluppo per evitare qualsiasi impatto involontario su altre macchine a stati. È possibile identificare la regola gestita specifica da eliminare esaminando il `Resource` campo utilizzato nel modello di policy per `events:PutRule` il servizio di destinazione. La regola gestita verrà ricreata la prossima volta che si crea o si aggiorna una macchina a stati che utilizza l'integrazione del servizio. Per ulteriori informazioni sull'eliminazione EventBridge delle regole, vedere Disabilitazione o eliminazione di una regola.
- Esegui il test con Step Functions Local, che non supporta l'uso di eventi per completare le attività Run a Job (.sync). Per utilizzare Step Functions Local, assumi il ruolo IAM utilizzato dalla tua macchina a stati. Potrebbe essere necessario modificare la relazione di fiducia. Imposta le variabili di `AWS_SESSION_TOKEN` ambiente `AWS_ACCESS_KEY_IDAWS_SECRET_ACCESS_KEY`, e sui valori del ruolo assunto, quindi avvia Step Functions Local utilizzandojava -jar StepFunctionsLocal.jar. Infine, usa il --endpoint-url parametro AWS CLI with the per creare una macchina a stati, avviare un'esecuzione e ottenere la cronologia delle esecuzioni. Per ulteriori informazioni, consulta Test di macchine a stato locale.

Se un'attività che utilizza il pattern Run a Job (.sync) viene interrotta, Step Functions farà il possibile per annullarla. Ciò richiede l'autorizzazione a `Cancel` `StopTerminate`, o azioni `Delete` API, come `batch:TerminateJob` o. `eks:DeleteCluster` Se nel tuo ruolo mancano queste autorizzazioni, Step Functions non sarà in grado di annullare l'attività e potresti addebitare costi aggiuntivi mentre questa continua a funzionare. Per ulteriori informazioni sull'interruzione delle attività, consulta Run a Job.

Modelli di policy utilizzati per creare ruoli IAM

I seguenti argomenti includono i modelli di policy utilizzati quando scegli che Step Functions crei un nuovo ruolo per te.

> **Note**
>
> Consulta questi modelli per capire come Step Functions crea le tue policy IAM e come esempio di come creare manualmente policy IAM per Step Functions quando lavori con

altri AWS servizi. Per ulteriori informazioni sulle integrazioni dei servizi Step Functions, vedereUtilizzo AWS Step Functions con altri servizi.

Argomenti

- Politiche IAM per Amazon API Gateway
- Politiche IAM per Amazon Athena
- Politiche IAM per AWS Batch
- IAM policies for Amazon Bedrock
- Politiche IAM per AWS CodeBuild
- Politiche IAM per Amazon DynamoDB
- Politiche IAM per Amazon ECS/AWS Fargate
- Politiche IAM per Amazon EKS
- Politiche IAM per Amazon EMR
- Politiche IAM per Amazon EMR su EKS
- Policy IAM per Amazon EMR Serverless
- Politiche IAM per Amazon EventBridge
- Politiche IAM per AWS Lambda
- Politiche IAM per AWS Elemental MediaConvert
- Politiche IAM per AWS Glue
- Politiche IAM per AWS Glue DataBrew
- Politiche IAM per Amazon SageMaker
- Politiche IAM per Amazon SNS
- Politiche IAM per Amazon SQS
- Politiche IAM per AWS Step Functions
- Politiche IAM per AWS X-Ray
- Attività o nessuna attività

Politiche IAM per Amazon API Gateway

I seguenti modelli di esempio mostrano come AWS Step Functions generare le politiche IAM in base alle risorse nella definizione della macchina a stati. Per ulteriori informazioni, consulta Politiche IAM per servizi integrati e Modelli di integrazione dei servizi.

Risorse:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "execute-api:Invoke"
            ],
            "Resource": [
                "arn:[[region]]:[[accountId]]:*"
            ]
        }
    ]
}
```

Il seguente esempio di codice mostra una politica delle risorse per chiamare 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>"
                    ]
                }
            }
```

```
        }
    ]
}
```

Politiche IAM per Amazon Athena

I seguenti modelli di esempio mostrano come AWS Step Functions generare le politiche IAM in base alle risorse nella definizione della macchina a stati. Per ulteriori informazioni, consulta Politiche IAM per servizi integrati e Modelli di integrazione dei servizi.

`StartQueryExecution`

Risorse statiche

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

Risorse dinamiche

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

Risorse

```
{
    "Version": "2012-10-17",
    "Statement":[
    {
        "Effect": "Allow",
        "Action": [
            "athena:stopQueryExecution"
        ],
        "Resource": [
            "arn:aws:athena:{{region}}:{{accountId}}:workgroup/*"
        ]
    }
    ]
}
```

GetQueryExecution

Risorse

```
{
    "Version": "2012-10-17",
    "Statement":[
    {
        "Effect": "Allow",
        "Action": [
```

```
            "athena:getQueryExecution"
        ],
        "Resource": [
            "arn:aws:athena:{{region}}:{{accountId}}:workgroup/*"
        ]
    }
    ]
}
```

`GetQueryResults`

Risorse

```
{
    "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:::*"
        ]
    }
    ]
}
```

Politiche IAM per AWS Batch

I seguenti modelli di esempio mostrano come AWS Step Functions generare le politiche IAM in base alle risorse nella definizione della macchina a stati. Per ulteriori informazioni, consulta Politiche IAM per servizi integrati e Modelli di integrazione dei servizi.

Poiché AWS Batch fornisce un supporto parziale per il controllo degli accessi a livello di risorsa, è necessario utilizzare. `"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

Quando si crea una macchina a stati utilizzando la console, crea Step Functions automaticamente un ruolo di esecuzione per la macchina a stati con i privilegi minimi richiesti. Questi IAM ruoli generati automaticamente sono validi per il momento Regione AWS in cui si crea la macchina a stati.

I seguenti modelli di esempio mostrano come AWS Step Functions generare le politiche IAM in base alle risorse nella definizione della macchina a stati. Per ulteriori informazioni, consulta Politiche IAM per servizi integrati e Modelli di integrazione dei servizi.

Quando crei delle IAM policy, ti consigliamo di non includere caratteri jolly nelle policy. Come best practice in materia di sicurezza, è consigliabile limitare il più possibile le policy. È consigliabile utilizzare politiche dinamiche solo quando determinati parametri di input non sono noti durante il runtime.

In questo argomento

- IAMesempi di policy per Amazon Bedrock l'integrazione con Step Functions

IAMesempi di policy per Amazon Bedrock l'integrazione con Step Functions

La sezione seguente descrive le IAM autorizzazioni necessarie in base all'Amazon BedrockAPI utilizzata per uno specifico modello di base o fornito. Questa sezione contiene anche esempi di politiche che garantiscono l'accesso completo.

Ricordati di sostituire il testo in *`corsivo`* con le informazioni specifiche della risorsa.

- IAMesempio di politica per accedere a un modello di base specifico utilizzando InvokeModel
- IAMesempio di policy per accedere a uno specifico modello predisposto utilizzando InvokeModel
- Esempio di IAM policy di accesso completo da utilizzare InvokeModel
- IAMesempio di policy per accedere a uno specifico modello di base come modello base
- IAMesempio di policy per accedere a un modello personalizzato specifico come modello base
- Esempio di IAM politica di accesso completo per utilizzare CreateModelCustomizationJob .sync

- IAMesempio di policy per accedere a un modello di base specifico utilizzando CreateModelCustomizationJob .sync
- IAMesempio di policy per accedere a un modello personalizzato utilizzando .sync CreateModelCustomizationJob
- Esempio di IAM politica di accesso completo per utilizzare .sync CreateModelCustomizationJob

IAMesempio di politica per accedere a un modello di base specifico utilizzando InvokeModel

Di seguito è riportato un esempio di IAM policy per una macchina a stati che accede a un modello di base specifico denominato `amazon.titan-text-express-v1` utilizzando l'azione 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"
            ]
        }
    ]
}
```

IAMesempio di policy per accedere a uno specifico modello predisposto utilizzando InvokeModel

Di seguito è riportato un esempio di IAM policy per una macchina a stati che accede a uno specifico modello fornito denominato `c2oi931ulksx` utilizzando l'InvokeModelazione API.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Sid": "InvokeModel1",
      "Action": [
```

```
        "bedrock:InvokeModel"
      ],
      "Resource": [
        "arn:aws:bedrock:us-east-2:123456789012:provisioned-model/c2oi931ulksx"
      ]
    }
  ]
}
```

Esempio di IAM policy di accesso completo da utilizzare InvokeModel

Di seguito è riportato un esempio di IAM policy per una macchina a stati che fornisce l'accesso completo quando si utilizza l'azione 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/*"
            ]
        }
    ]
}
```

IAMesempio di policy per accedere a uno specifico modello di base come modello base

Di seguito è riportato un esempio di IAM policy per una macchina a stati per accedere a un modello di base specifico `amazon.titan-text-express-v1` denominato modello base utilizzando l'azione 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"
            ]
        }
    ]
}
```

IAMesempio di policy per accedere a un modello personalizzato specifico come modello base

Di seguito è riportato un esempio di IAM policy per una macchina a stati per accedere a un modello personalizzato specifico come modello base utilizzando l'azione 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]]"
            ]
        }
    ]
}
```

Esempio di IAM politica di accesso completo per utilizzare CreateModelCustomizationJob .sync

Di seguito è riportato un esempio di IAM policy per una macchina a stati che fornisce l'accesso completo quando si utilizza l'azione 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"
            ]
        }
    ]
```

```
}
```

IAMesempio di policy per accedere a un modello di base specifico utilizzando CreateModelCustomizationJob .sync

Di seguito è riportato un esempio di IAM policy per una macchina a stati per accedere a un modello di base specifico denominato `amazon.titan-text-express-v1` utilizzando l'azione CreateModelCustomizationJobAPI.sync.

```
{
    "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"
            ]
        }
    ]
}
```

IAMesempio di policy per accedere a un modello personalizzato utilizzando .sync CreateModelCustomizationJob

Di seguito è riportato un esempio di IAM policy per una macchina a stati per accedere a un modello personalizzato utilizzando l'azione API CreateModelCustomizationJob.sync.

```
{
    "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"
            ]
        }
    ]
}
```

Esempio di IAM politica di accesso completo per utilizzare .sync CreateModelCustomizationJob

Di seguito è riportato un esempio di IAM policy per una macchina a stati che fornisce l'accesso completo quando si utilizza l'azione CreateModelCustomizationJobAPI.sync.

```
{
    "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"
            ]
        }
    ]
}
```

Politiche IAM per AWS CodeBuild

I seguenti modelli di esempio mostrano come AWS Step Functions generare le politiche IAM in base alle risorse nella definizione della macchina a stati. Per ulteriori informazioni, consulta Politiche IAM per servizi integrati e Modelli di integrazione dei servizi.

Risorse:

```
{
    "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

Risorse statiche

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]]"
      ]
    }
  ]
}
```

Risorse dinamiche

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`

Risorse statiche

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:StopBuild"
      ],
```

```
      "Resource": [
        "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
      ]
    }
  ]
}
```

Risorse dinamiche

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:StopBuild"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:*:project/*"
      ]
    }
  ]
}
```

`BatchDeleteBuilds`

Risorse statiche

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:BatchDeleteBuilds"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
      ]
    }
  ]
}
```

Risorse dinamiche

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:BatchDeleteBuilds"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:*:project/*"
      ]
    }
  ]
}
```

`BatchGetReports`

Risorse statiche

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:BatchGetReports"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:[[accountId]]:report-group/[[reportName]]"
      ]
    }
  ]
}
```

Risorse dinamiche

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": [
        "codebuild:BatchGetReports"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:*:report-group/*"
      ]
    }
  ]
}
```

StartBuildBatch

Risorse statiche

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]]"
            ]
        }
    ]
}
```

Risorse dinamiche

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`

Risorse statiche

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:StopBuildBatch"
            ],
            "Resource": [
```

```
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
            ]
        }
    ]
}
```

Risorse dinamiche

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:StopBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/*"
            ]
        }
    ]
}
```

`RetryBuildBatch`

Risorse statiche

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]]"
            ]
        }
    ]
}
```

Risorse dinamiche

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`

Risorse statiche

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:DeleteBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
            ]
        }
    ]
}
```

Risorse dinamiche

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:DeleteBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/*"
            ]
        }
    ]
}
```

Politiche IAM per Amazon DynamoDB

I seguenti modelli di esempio mostrano come AWS Step Functions generare le politiche IAM in base alle risorse nella definizione della macchina a stati. Per ulteriori informazioni, consulta Politiche IAM per servizi integrati e Modelli di integrazione dei servizi.

Risorse statiche

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:[[region]]:[[accountId]]:table/[[tableName]]"
            ]
        }
    ]
}
```

Risorse dinamiche

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem"
            ],
            "Resource": "*"
        }
    ]
}
```

Per ulteriori informazioni sulle policy IAM per tutte le azioni API DynamoDB, consulta le politiche IAM con DynamoDB nella Amazon DynamoDB Developer Guide. Inoltre, per informazioni sulle policy IAM per PartiQL for DynamoDB, consulta Policies IAM with PartiQL for DynamoDB nella Amazon DynamoDB Developer Guide.

Politiche IAM per Amazon ECS/AWS Fargate

I seguenti modelli di esempio mostrano come AWS Step Functions generare le politiche IAM in base alle risorse nella definizione della macchina a stati. Per ulteriori informazioni, consulta Politiche IAM per servizi integrati e Modelli di integrazione dei servizi.

Poiché il valore di non `TaskId` è noto fino all'invio dell'attività, Step Functions crea una `"Resource": "*"` politica più privilegiata.

> **Note**
>
> Puoi interrompere solo le attività di Amazon Elastic Container Service (Amazon ECS) avviate da Step Functions, nonostante `"*"` la policy IAM.

Run a Job (.sync)

Risorse statiche

```
{
```

```
    "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"
            ]
        }
    ]
}
```

Risorse dinamiche

```
{
    "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)

Risorse statiche

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RunTask"
            ],
            "Resource": [
                "arn:aws:ecs:[[region]]:
[[accountId]]:task-definition/[[taskDefinition]]"
            ]
        }
    ]
}
```

Risorse dinamiche

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RunTask"
            ],
            "Resource": "*"
        }
    ]
}
```

Se le attività pianificate di Amazon ECS richiedono l'uso di un ruolo di esecuzione di attività, un ruolo di attività o un'eccezione di ruolo dell'attività, è necessario aggiungere `iam:PassRole` le autorizzazioni per ogni ruolo di esecuzione dell'attività, ruolo dell'attività o sovrascrittura del ruolo dell'attività al ruolo IAM CloudWatch Events dell'entità chiamante, che in questo caso è Step Functions.

Politiche IAM per Amazon EKS

I seguenti modelli di esempio mostrano come AWS Step Functions generare le politiche IAM in base alle risorse nella definizione della macchina a stati. Per ulteriori informazioni, consulta Politiche IAM per servizi integrati e Modelli di integrazione dei servizi.

`CreateCluster`

Risorse

```
{
    "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`

Risorse

```
{
    "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`

Risorse

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DeleteCluster",
                "eks:DescribeCluster"
            ],
            "Resource": [
```

```
                "arn:aws:eks:sa-east-1:444455556666:cluster/ExampleCluster"
            ]
        }
    ]
}
```

`DeleteNodegroup`

Risorse

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DeleteNodegroup",
                "eks:DescribeNodegroup"
            ],
            "Resource": [
                "arn:aws:eks:sa-east-1:444455556666:nodegroup/ExampleCluster/
ExampleNodegroup/*"
            ]
        }
    ]
}
```

Per ulteriori informazioni sull'uso di Amazon EKS con Step Functions, consultaChiama Amazon EKS con Step Functions.

Politiche IAM per Amazon EMR

I seguenti modelli di esempio mostrano come AWS Step Functions generare le politiche IAM in base alle risorse nella definizione della macchina a stati. Per ulteriori informazioni, consulta Politiche IAM per servizi integrati e Modelli di integrazione dei servizi.

`addStep`

Risorse statiche

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:AddJobFlowSteps",
        "elasticmapreduce:DescribeStep",
        "elasticmapreduce:CancelSteps"
      ],
      "Resource": [
        "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
      ]
    }
  ]
}
```

Risorse dinamiche

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:AddJobFlowSteps",
        "elasticmapreduce:DescribeStep",
        "elasticmapreduce:CancelSteps"
      ],
      "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
    }
  ]
}
```

`cancelStep`

Risorse statiche

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticmapreduce:CancelSteps",
```

```
            "Resource": [

 "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
            ]
        }
    ]
}
```

Risorse dinamiche

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticmapreduce:CancelSteps",
            "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
        }
    ]
}
```

createCluster

Risorse statiche

```
{
  "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`

Risorse statiche

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticmapreduce:SetTerminationProtection",
            "Resource": [

 "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
            ]
        }
    ]
}
```

Risorse dinamiche

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticmapreduce:SetTerminationProtection",
            "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
        }
    ]
}
```

`modifyInstanceFleetByName`

Risorse statiche

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:ModifyInstanceFleet",
                "elasticmapreduce:ListInstanceFleets"
            ],
            "Resource": [

 "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
            ]
        }
    ]
}
```

Risorse dinamiche

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:ModifyInstanceFleet",
                "elasticmapreduce:ListInstanceFleets"
            ],
            "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
        }
    ]
}
```

`modifyInstanceGroupByName`

Risorse statiche

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:ModifyInstanceGroups",
                "elasticmapreduce:ListInstanceGroups"
```

```
            ],
            "Resource": [

 "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
            ]
        }
    ]
}
```

Risorse dinamiche

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:ModifyInstanceGroups",
                "elasticmapreduce:ListInstanceGroups"
            ],
            "Resource": "*"
        }
    ]
}
```

`terminateCluster`

Risorse statiche

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:TerminateJobFlows",
        "elasticmapreduce:DescribeCluster"
      ],
      "Resource": [
        "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
      ]
    }
  ]
```

```
}
```

Risorse dinamiche

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:TerminateJobFlows",
        "elasticmapreduce:DescribeCluster"
      ],
      "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
    }
  ]
}
```

Politiche IAM per Amazon EMR su EKS

I seguenti modelli di esempio mostrano come AWS Step Functions generare le politiche IAM in base alle risorse nella definizione della macchina a stati. Per ulteriori informazioni, consulta Politiche IAM per servizi integrati e Modelli di integrazione dei servizi.

`CreateVirtualCluster`

Risorse

```
{
    "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`

Risorse statiche

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]]"
      ]
    }
  ]
}
```

Risorse dinamiche

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`

Risorse statiche

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]]"
          ]
        }
      }
    }
  ]
}
```

Risorse dinamiche

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]]"
          ]
        }
      }
    }
  ]
}
```

Policy IAM per Amazon EMR Serverless

Quando crei una macchina a stati utilizzando la console, crea Step Functions automaticamente un ruolo di esecuzione per la tua macchina a stati con i privilegi minimi richiesti. Questi IAM ruoli generati automaticamente sono validi per il momento Regione AWS in cui si crea la macchina a stati.

I seguenti modelli di esempio mostrano come AWS Step Functions generare le politiche IAM in base alle risorse nella definizione della macchina a stati. Per ulteriori informazioni, consulta Politiche IAM per servizi integrati e Modelli di integrazione dei servizi.

Quando crei delle IAM policy, ti consigliamo di non includere caratteri jolly nelle policy. Come best practice in materia di sicurezza, è consigliabile limitare il più possibile le policy. È consigliabile utilizzare politiche dinamiche solo quando determinati parametri di input non sono noti durante il runtime.

Inoltre, gli utenti amministratori devono prestare attenzione quando concedono a utenti non amministratori ruoli di esecuzione per l'esecuzione delle macchine a stati. Ti consigliamo di includere le politiche PassRole nei ruoli di esecuzione se stai creando politiche da solo. Ti consigliamo inoltre di aggiungere le chiavi `aws:SourceARN` e di `aws:SourceAccount` contesto nei ruoli di esecuzione.

In questo argomento

- Esempi di policy IAM per l'integrazione EMR Serverless con Step Functions

Esempi di policy IAM per l'integrazione EMR Serverless con Step Functions

- Esempio di policy IAM per CreateApplication
- Esempio di policy IAM per StartApplication
- Esempio di policy IAM per StopApplication
- Esempio di policy IAM per DeleteApplication
- Esempio di policy IAM per StartJobRun
- Esempio di policy IAM per CancelJobRun

Esempio di policy IAM per CreateApplication

Di seguito è riportato un esempio di policy IAM per una macchina a stati con uno CreateApplication Attività stato.

> **Note**
>
> È necessario specificare le CreateServiceLinkedRole autorizzazioni nelle politiche IAM durante la creazione della prima applicazione in assoluto nel proprio account. Successivamente, non è necessario aggiungere questa autorizzazione. Per informazioni

CreateServiceLinkedRolein merito CreateServiceLinkedRole, vedere https://docs.aws.amazon.com/IAM/latest/APIReference/.

Le risorse statiche e dinamiche per le seguenti politiche sono le stesse.

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"
                }
            }
        }
    ]
}
```

Esempio di policy IAM per StartApplication

Risorse statiche

Di seguito sono riportati esempi di policy IAM per risorse statiche quando si utilizza una macchina a stati con uno StartApplication Attività stato.

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]]"
            ]
        }
    ]
}
```

Risorse dinamiche

Di seguito sono riportati alcuni esempi di policy IAM per risorse dinamiche quando si utilizza una macchina a stati con uno StartApplication Attività stato.

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/*"
            ]
        }
    ]
}
```

Esempio di policy IAM per StopApplication

Risorse statiche

Di seguito sono riportati esempi di policy IAM per risorse statiche quando si utilizza una macchina a stati con uno StopApplication Attività stato.

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]]"
            ]
        }
    ]
}
```

Risorse dinamiche

Di seguito sono riportati alcuni esempi di policy IAM per risorse dinamiche quando si utilizza una macchina a stati con uno StopApplication Attività stato.

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/*"
            ]
        }
    ]
}
```

Esempio di policy IAM per DeleteApplication

Risorse statiche

Di seguito sono riportati esempi di policy IAM per risorse statiche quando si utilizza una macchina a stati con uno DeleteApplication Attività stato.

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]]"
            ]
        }
    ]
}
```

Risorse dinamiche

Di seguito sono riportati alcuni esempi di policy IAM per risorse dinamiche quando si utilizza una macchina a stati con uno DeleteApplication Attività stato.

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/*"
            ]
        }
    ]
}
```

Esempio di policy IAM per StartJobRun

Risorse statiche

Di seguito sono riportati esempi di policy IAM per risorse statiche quando si utilizza una macchina a stati con uno StartJobRun Attività stato.

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"
               }
           }
       }
    ]
}
```

Risorse dinamiche

Di seguito sono riportati alcuni esempi di policy IAM per risorse dinamiche quando si utilizza una macchina a stati con uno StartJobRun Attività stato.

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"
                }
            }
        }
    ]
}
```

Esempio di policy IAM per CancelJobRun

Risorse statiche

Di seguito sono riportati esempi di policy IAM per risorse statiche quando si utilizza una macchina a stati con uno CancelJobRun Attività stato.

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]]"
            ]
        }
    ]
}
```

Risorse dinamiche

Di seguito sono riportati alcuni esempi di policy IAM per risorse dinamiche quando si utilizza una macchina a stati con uno CancelJobRun Attività stato.

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/*"
            ]
        }
    ]
}
```

Politiche IAM per Amazon EventBridge

I seguenti modelli di esempio mostrano come AWS Step Functions generare le politiche IAM in base alle risorse nella definizione della macchina a stati. Per ulteriori informazioni, consulta Politiche IAM per servizi integrati e Modelli di integrazione dei servizi.

`PutEvents`

Risorse statiche

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "events:PutEvents"
            ],
            "Resource": [
                "arn:aws:events:us-east-1:123456789012:event-bus/stepfunctions-
sampleproject-eventbus"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Risorse dinamiche

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "events:PutEvents"
            ],
            "Resource": "arn:aws:events:*:*:event-bus/*"
        }
    ]
}
```

Per ulteriori informazioni sull'utilizzo EventBridge con Step Functions, vedereChiama EventBridge con Step Functions.

Politiche IAM per AWS Lambda

I seguenti modelli di esempio mostrano come AWS Step Functions generare le politiche IAM in base alle risorse nella definizione della macchina a stati. Per ulteriori informazioni, consulta Politiche IAM per servizi integrati e Modelli di integrazione dei servizi.

AWS Step Functions genera una policy IAM basata sulla definizione della macchina a stati. Per una macchina a stati con due stati di AWS Lambda attività che chiamano `function1` `efunction2`, è necessario utilizzare una policy con `lambda:Invoke` autorizzazioni per le due funzioni.

Questo viene mostrato nell'esempio seguente.

```
{
    "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]]"
            ]
        }
    ]
}
```

Politiche IAM per AWS Elemental MediaConvert

I seguenti modelli di esempio mostrano come sia AWS Step Functions necessario configurare le politiche IAM in base alle risorse nella definizione della macchina a stati. Puoi utilizzare la console IAM per aggiungere eventuali policy relative ai ruoli mancanti. Per ulteriori informazioni, consulta Politiche IAM per servizi integrati e Modelli di integrazione dei servizi.

Poiché MediaConvert fornisce un supporto parziale per il controllo degli accessi a livello di risorsa, è necessario utilizzare. `"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": "*"
        }
    ]
}
```

Politiche IAM per AWS Glue

I seguenti modelli di esempio mostrano come AWS Step Functions generare le politiche IAM in base alle risorse nella definizione della macchina a stati. Per ulteriori informazioni, consulta Politiche IAM per servizi integrati e Modelli di integrazione dei servizi.

AWS Glue non ha un controllo basato sulle risorse.

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": "*"
        }
    ]
}
```

Politiche IAM per AWS Glue DataBrew

I seguenti modelli di esempio mostrano come AWS Step Functions generare le politiche IAM in base alle risorse nella definizione della macchina a stati. Per ulteriori informazioni, consulta Politiche IAM per servizi integrati e Modelli di integrazione dei servizi.

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/*"
            ]
        }
    ]
}
```

Politiche IAM per Amazon SageMaker

I seguenti modelli di esempio mostrano come AWS Step Functions generare le politiche IAM in base alle risorse nella definizione della macchina a stati. Per ulteriori informazioni, consulta Politiche IAM per servizi integrati e Modelli di integrazione dei servizi.

> **Note**
>
> Per questi esempi, fai *[[roleArn]]* riferimento all'Amazon Resource Name (ARN) del ruolo IAM che SageMaker utilizza per accedere agli artefatti del modello e alle immagini docker per la distribuzione su istanze di calcolo ML o per lavori di trasformazione in batch. Per ulteriori informazioni, consulta Amazon SageMaker Roles.

`CreateTrainingJob`

Risorse statiche

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"
        }
      }
    }
  ]
}
```

Risorse dinamiche

.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 non creerà automaticamente una policy per `CreateTransformJob` quando crei una macchina a stati che si integra con SageMaker. È necessario allegare una policy in linea al ruolo creato sulla base di uno dei seguenti esempi IAM.

Risorse statiche

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"
        }
      }
    }
  ]
```

```
}
```

Risorse dinamiche

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"
        }
      }
    }
  ]
}
```

Politiche IAM per Amazon SNS

I seguenti modelli di esempio mostrano come AWS Step Functions generare le politiche IAM in base alle risorse nella definizione della macchina a stati. Per ulteriori informazioni, consulta Politiche IAM per servizi integrati e Modelli di integrazione dei servizi.

Risorse statiche

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:[[region]]:[[accountId]]:[[topicName]]"
            ]
        }
    ]
}
```

Risorse basate su un percorso o che pubblicano in *TargetArn* o *PhoneNumber*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": "*"
        }
    ]
}
```

Politiche IAM per Amazon SQS

I seguenti modelli di esempio mostrano come AWS Step Functions generare le politiche IAM in base alle risorse nella definizione della macchina a stati. Per ulteriori informazioni, consulta Politiche IAM per servizi integrati e Modelli di integrazione dei servizi.

Risorse statiche

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sqs:SendMessage"
            ],
            "Resource": [
                "arn:aws:sqs:[[region]]:[[accountId]]:[[queueName]]"
            ]
        }
    ]
}
```

Risorse dinamiche

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sqs:SendMessage"
            ],
            "Resource": "*"
        }
    ]
}
```

Politiche IAM per AWS Step Functions

Per una macchina a stati che richiede `StartExecution` l'esecuzione di un singolo flusso di lavoro annidato, utilizza una policy IAM che limiti le autorizzazioni a quella macchina a stati.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "states:StartExecution"
            ],
            "Resource": [

 "arn:aws:states:[[region]]:[[accountId]]:stateMachine:[[stateMachineName]]"
            ]
        }
    ]
}
```

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Utilizzo AWS Step Functions con altri servizi
- Passa i parametri a un'API di servizio
- Gestisci AWS Step Functions le esecuzioni come servizio integrato

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]]"
            ]
        }
    ]
}
```

Per ulteriori informazioni sulle esecuzioni di flussi di lavoro nidificati, consulta Avvio delle esecuzioni di flusso di lavoro da uno stato di task.

Politiche IAM per AWS X-Ray

I seguenti modelli di esempio mostrano come AWS Step Functions generare le politiche IAM in base alle risorse nella definizione della macchina a stati. Per ulteriori informazioni, consulta Politiche IAM per servizi integrati e Modelli di integrazione dei servizi.

Per abilitare il tracciamento X-Ray, avrai bisogno di una policy IAM con autorizzazioni adeguate per consentire il tracciamento. Se la tua macchina a stati utilizza altri servizi integrati, potresti aver bisogno di politiche IAM aggiuntive. Consulta le politiche IAM per le integrazioni di servizi specifiche.

Quando si crea una macchina a stati con il tracciamento X-Ray abilitato, viene creata automaticamente una policy IAM.

> **Note**
>
> Se si abilita il tracciamento a raggi X per una macchina a stati esistente, è necessario assicurarsi di aggiungere una politica con autorizzazioni sufficienti per abilitare le tracce a raggi X.

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

```
        {
            "Effect": "Allow",
            "Action": [
                "xray:PutTraceSegments",
                "xray:PutTelemetryRecords",
                "xray:GetSamplingRules",
                "xray:GetSamplingTargets"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Per ulteriori informazioni sull'utilizzo di X-Ray with Step Functions, vedere. AWS X-Ray e Step Functions

Attività o nessuna attività

Per una macchina a stati che ha solo `Activity` attività o nessuna attività, utilizza una policy IAM che neghi l'accesso a tutte le azioni e le risorse.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "*",
            "Resource": "*"
        }
    ]
}
```

Per ulteriori informazioni sull'utilizzo delle attività `Activity` , consulta Attività.

Politiche IAM per l'utilizzo dello stato della mappa distribuita

Quando crei flussi di lavoro con la console Step Functions, Step Functions può generare automaticamente policy IAM basate sulle risorse nella definizione del flusso di lavoro. Queste politiche includono i privilegi minimi necessari per consentire al ruolo della macchina a stati di richiamare l'azione dell'`StartExecution`API per lo stato della mappa distribuita. Queste policy

includono anche i privilegi minimi necessari Step Functions per accedere alle AWS risorse, come i bucket e gli oggetti Amazon S3 e le funzioni Lambda. Ti consigliamo vivamente di includere solo le autorizzazioni necessarie nelle tue policy IAM. Ad esempio, se il tuo flusso di lavoro include uno `Map` stato in modalità Distribuita, limita le policy allo specifico bucket e alla cartella Amazon S3 che contiene il set di dati.

> **Important**
>
> Se specifichi un bucket Amazon S3 e un oggetto, o prefisso, con un percorso di riferimento a una coppia chiave-valore esistente nell'input dello stato della mappa distribuita, assicurati di aggiornare le policy IAM per il tuo flusso di lavoro. Definisci le policy fino ai nomi dei bucket e degli oggetti in cui il percorso si risolve in fase di esecuzione.

In questo argomento:

- Esempio di policy IAM per l'esecuzione di uno stato della mappa distribuita
- Esempio di policy IAM per redriving una mappa distribuita
- Esempi di policy IAM per la lettura dei dati dai set di dati di Amazon S3
- Esempio di policy IAM per la scrittura di dati in un bucket Amazon S3

Esempio di policy IAM per l'esecuzione di uno stato della mappa distribuita

Quando includi uno stato della mappa distribuita nei flussi di lavoro, Step Functions necessita delle autorizzazioni appropriate per consentire al ruolo della macchina a stati di richiamare l'azione `StartExecution` API per lo stato della mappa distribuita.

Il seguente esempio di policy IAM concede i privilegi minimi richiesti al ruolo della macchina a stati per l'esecuzione dello stato della mappa distribuita.

> **Note**
>
> Assicurati di sostituirlo *`stateMachineName`* con il nome della macchina a stati in cui stai utilizzando lo stato Distributed Map. Ad esempio, `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:*"
    }
  ]
}
```

Esempio di policy IAM per redriving una mappa distribuita

È possibile riavviare le esecuzioni di workflow secondarie non riuscite in una mappa eseguita dal redrivingflusso di lavoro principale. Un flusso di lavoro redriven principale con redrives tutti gli stati non riusciti, inclusa Distributed Map. Assicurati che il tuo ruolo di esecuzione disponga dei privilegi minimi necessari per consentirgli di richiamare l'azione `RedriveExecution` API sul flusso di lavoro principale.

Il seguente esempio di policy IAM concede i privilegi minimi richiesti al ruolo di macchina a stati per redriving uno stato di Distributed Map.

> **Note**
>
> Assicurati di sostituirlo *stateMachineName* con il nome della macchina a stati in cui stai utilizzando lo stato Distributed Map. Ad esempio, `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:*"
    }
  ]
}
```

Esempi di policy IAM per la lettura dei dati dai set di dati di Amazon S3

I seguenti esempi di policy IAM garantiscono i privilegi minimi necessari per accedere ai set di dati Amazon S3 utilizzando ListObjectsIe azioni GetObjectV2 e API.

Example Policy IAM per oggetti Amazon S3 come set di dati

L'esempio seguente mostra una policy IAM che concede i privilegi minimi per accedere agli oggetti organizzati all'interno di un *processImages* bucket Amazon S3 denominato. *myBucket*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket"
            ],
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
                        "processImages"
                    ]
                }
            }
        }
    ]
```

```
}
```

Example Politica IAM per un file CSV come set di dati

L'esempio seguente mostra una policy IAM che concede i privilegi minimi per accedere a un file CSV denominato. *ratings.csv*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket/csvDataset/ratings.csv"
            ]
        }
    ]
}
```

Example Policy IAM per un inventario Amazon S3 come set di dati

L'esempio seguente mostra una policy IAM che concede i privilegi minimi per accedere a un report di inventario di 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/*"
            ]
        }
    ]
```

```
}
```

Esempio di policy IAM per la scrittura di dati in un bucket Amazon S3

Il seguente esempio di policy IAM concede i privilegi minimi richiesti per scrivere i risultati dell'esecuzione del flusso di lavoro secondario in una cartella denominata *CSVJobs in un bucket* Amazon S3 utilizzando l'azione API. PutObject

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
            "Resource": [
                "arn:aws:s3:::resultBucket/csvJobs/*"
            ]
        }
    ]
}
```

Autorizzazioni IAM per bucket Amazon S3 AWS KMS key crittografato

Lo stato della mappa distribuito utilizza caricamenti in più parti per scrivere i risultati dell'esecuzione del flusso di lavoro secondario in un bucket Amazon S3. Se il bucket è crittografato utilizzando una AWS Key Management Service (AWS KMS) chiave, devi includere anche le autorizzazioni per eseguire la IAM chiave e le `kms:Decrypt` azioni sulla chiave. `kms:Encrypt` `kms:GenerateDataKey` Queste autorizzazioni sono obbligatorie perché Amazon S3 deve decrittografare e leggere i dati dalle parti di file crittografate prima di completare il caricamento in più parti.

Il seguente esempio di policy IAM concede l'`kms:Decrypt`autorizzazione e `kms:Encrypt` le `kms:GenerateDataKey` azioni sulla chiave utilizzata per crittografare il bucket Amazon S3.

```
{
  "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"
    ]
  }
}
```

Per ulteriori informazioni, consulta Caricamento di un file di grandi dimensioni su Amazon S3 con la crittografia utilizzando una AWS KMS key nel Knowledge Center di AWS .

Se il tuo utente o ruolo IAM è lo Account AWS stesso diKMS key, devi disporre di queste autorizzazioni sulla policy chiave. Se il tuo utente o ruolo IAM appartiene a un account diverso da quelloKMS key, devi disporre delle autorizzazioni sia sulla policy chiave che sul tuo utente o ruolo IAM.

Policy basate su tag

Step Functions supporta le politiche basate sui tag. Ad esempio, è possibile limitare l'accesso a tutte le risorse Step Functions che includono un tag con la chiave `environment` e il valore`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"}
            }
```

```
        }
    ]
}
```

Questa policy `Deny` (Nega) la possibilità di eliminare lo stato di attività o di macchine, arrestare esecuzioni e aggiungere o eliminare i nuovi tag per tutte le risorse che sono state contrassegnate come `environment/production`.

Per l'autorizzazione basata su tag, le risorse di esecuzione di una macchina a stati, come illustrato nell'esempio seguente, ereditano i tag associati a una macchina a stati.

```
arn:<partition>:states:<Region>:<account-id>:execution:<StateMachineName>:<ExecutionId>
```

Quando chiamate DescribeExecutiono altre API in cui specificate la risorsa di esecuzione ARN, Step Functions utilizza i tag associati alla macchina a stati per accettare o rifiutare la richiesta durante l'esecuzione dell'autorizzazione basata su tag. Ciò consente di consentire o negare l'accesso alle esecuzioni delle macchine a stati a livello di macchina a stati.

Per ulteriori informazioni sul tagging, consulta:

- Inserimento di tag in Step Functions
- Controllo degli accessi tramite tag IAM

Risoluzione dei problemi di AWS Step Functions identità e accesso

Utilizza le seguenti informazioni per aiutarti a diagnosticare e risolvere i problemi più comuni che potresti riscontrare quando lavori con Step Functions e IAM.

Argomenti

- Non sono autorizzato a eseguire un'azione in Step Functions
- Non sono autorizzato a eseguire iam: PassRole
- Voglio consentire a persone esterne a me di accedere Account AWS alle mie risorse Step Functions

Non sono autorizzato a eseguire un'azione in Step Functions

Se ricevi un errore che indica che non disponi dell'autorizzazione per eseguire un'operazione, le tue policy devono essere aggiornate in modo che ti sei consentito eseguire tale operazione.

Il seguente esempio di errore si verifica quando l'utente `mateojackson` prova a utilizzare la console per visualizzare i dettagli relativi a una risorsa *`my-example-widget`* fittizia, ma non dispone di autorizzazioni `states:`*`GetWidget`* fittizie.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 states:GetWidget on resource: my-example-widget
```

In questo caso, la policy deve essere aggiornata in modo che Mateo possa accedere alla risorsa *`my-example-widget`* mediante l'operazione `states:`*`GetWidget`*.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Non sono autorizzato a eseguire iam: PassRole

Se ricevi un messaggio di errore indicante che non sei autorizzato a eseguire l'`iam:PassRole`azione, le tue politiche devono essere aggiornate per consentirti di trasferire un ruolo a Step Functions.

Alcuni Servizi AWS consentono di trasferire un ruolo esistente a quel servizio invece di creare un nuovo ruolo di servizio o un ruolo collegato al servizio. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per trasmettere il ruolo al servizio.

Il seguente errore di esempio si verifica quando un utente IAM denominato `marymajor` tenta di utilizzare la console per eseguire un'azione in Step Functions. Tuttavia, l'azione richiede che il servizio disponga delle autorizzazioni concesse da un ruolo di servizio. Mary non dispone delle autorizzazioni per passare il ruolo al servizio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In questo caso, le policy di Mary devono essere aggiornate per poter eseguire l'operazione `iam:PassRole`.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Voglio consentire a persone esterne a me di accedere Account AWS alle mie risorse Step Functions

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per servizi che supportano policy basate su risorse o liste di controllo accessi (ACL), utilizza tali policy per concedere alle persone l'accesso alle tue risorse.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Per sapere se Step Functions supporta queste funzionalità, vedereCome AWS Step Functions funziona con IAM.
- Per scoprire come fornire l'accesso alle tue risorse su tutto Account AWS ciò che possiedi, consulta Fornire l'accesso a un utente IAM in un altro Account AWS di tua proprietà nella IAM User Guide.
- Per scoprire come fornire l'accesso alle tue risorse a terze parti Account AWS, consulta Fornire l'accesso a soggetti Account AWS di proprietà di terze parti nella Guida per l'utente IAM.
- Per informazioni su come fornire l'accesso tramite la federazione delle identità, consulta Fornire l'accesso a utenti autenticati esternamente (Federazione delle identità) nella Guida per l'utente di IAM.
- Per informazioni sulle differenze tra l'utilizzo di ruoli e policy basate su risorse per l'accesso multi-account, consulta Differenza tra i ruoli IAM e le policy basate su risorse nella Guida per l'utente IAM.

Registrazione e monitoraggio

Per informazioni sulla registrazione e il monitoraggio AWS Step Functions, consulta la Registrazione e monitoraggio sezione.

Convalida della conformità per AWS Step Functions

I revisori esterni valutano la sicurezza e la conformità nell' AWS Step Functions ambito di più programmi di AWS conformità. Questi includono SOC, PCI, FedRAMP, HIPAA e altri.

Per un elenco dei AWS servizi che rientrano nell'ambito di specifici programmi di conformità, vedere AWS Servizi nell'ambito del programma di conformitàAWS . Per informazioni generali, vedere Programmi di AWS conformità Programmi di di .

È possibile scaricare report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta Scaricamento dei report in AWS Artifact .

La tua responsabilità di conformità quando utilizzi Step Functions è determinata dalla sensibilità dei tuoi dati, dagli obiettivi di conformità della tua azienda e dalle leggi e dai regolamenti applicabili. AWS fornisce le seguenti risorse per contribuire alla conformità:

- Guide rapide su sicurezza e conformità Guide introduttive implementazione illustrano considerazioni sull'architettura e forniscono passaggi per implementare ambienti di base incentrati sulla sicurezza e la conformità. AWS
- Progettazione per la sicurezza e la conformità HIPAA su Amazon Web Services: questo white paper descrive come le aziende possono utilizzare AWS per creare applicazioni conformi allo standard HIPAA.
- AWS Risorse per la conformità Risorse per AWS : questa raccolta di cartelle di lavoro e guide potrebbe riguardare il tuo settore e la tua area geografica.
- Valutazione delle risorse con le regole nella Guida per gli AWS Config sviluppatori: il AWS Config servizio valuta la conformità delle configurazioni delle risorse alle pratiche interne, alle linee guida del settore e alle normative.
- AWS Security Hub— Questo AWS servizio offre una visione completa dello stato di sicurezza dell'utente e consente di verificare la conformità agli standard e alle best practice del settore della sicurezza. AWS

Resilienza in AWS Step Functions

L'infrastruttura AWS globale è costruita attorno a AWS regioni e zone di disponibilità. AWS Le regioni forniscono più zone di disponibilità fisicamente separate e isolate, collegate con reti a bassa latenza, ad alto throughput e altamente ridondanti. Con le zone di disponibilità, puoi progettare e gestire applicazioni e database che eseguono automaticamente il failover tra zone di disponibilità senza interruzioni. Le zone di disponibilità sono più disponibili, tolleranti ai guasti e scalabili rispetto alle infrastrutture a data center singolo o multiplo tradizionali.

Per ulteriori informazioni su AWS regioni e zone di disponibilità, consulta Global Infrastructure.AWS

Oltre all'infrastruttura AWS globale, Step Functions offre diverse funzionalità per supportare le esigenze di resilienza e backup dei dati.

Sicurezza dell'infrastruttura in AWS Step Functions

In quanto servizio gestito, AWS Step Functions è protetto dalla sicurezza di rete AWS globale. Per informazioni sui servizi AWS di sicurezza e su come AWS protegge l'infrastruttura, consulta AWS Cloud Security. Per progettare il tuo AWS ambiente utilizzando le migliori pratiche per la sicurezza dell'infrastruttura, vedi Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Utilizzi chiamate API AWS pubblicate per accedere a Step Functions attraverso la rete. I client devono supportare quanto segue:

- Transport Layer Security (TLS). È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Suite di cifratura con Perfect Forward Secrecy (PFS), ad esempio Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Inoltre, le richieste devono essere firmate utilizzando un ID chiave di accesso e una chiave di accesso segreta associata a un principale IAM. O puoi utilizzare AWS Security Token Service (AWS STS) per generare credenziali di sicurezza temporanee per sottoscrivere le richieste.

Puoi chiamare le operazioni AWS API da qualsiasi posizione di rete, ma Step Functions non supporta le politiche di accesso basate sulle risorse, che possono includere restrizioni basate sull'indirizzo IP di origine. Inoltre, puoi utilizzare le policy Step Functions per controllare l'accesso da endpoint Amazon Virtual Private Cloud (Amazon VPC) specifici o VPC specifici. In effetti, questo isola l'accesso alla rete a una determinata Step Functions risorsa solo dal VPC specifico all'interno AWS della rete.

Analisi della configurazione e delle vulnerabilità in AWS Step Functions

La configurazione e i controlli IT sono una responsabilità condivisa tra te AWS e te, nostro cliente. Per ulteriori informazioni, consulta il modello di responsabilità AWS condivisa.

Migrazione dei carichi di lavoro da Step AWS Data Pipeline Functions

AWS ha lanciato il AWS Data Pipeline servizio nel 2012. A quel tempo, i clienti desideravano un servizio che consentisse loro di utilizzare una varietà di opzioni di elaborazione per spostare i dati tra diverse fonti di dati. Man mano che le esigenze di trasferimento dei dati sono cambiate nel tempo, sono cambiate anche le soluzioni a tali esigenze. Ora avete la possibilità di scegliere la soluzione che più si avvicina alle vostre esigenze aziendali. Ad esempio, puoi eseguire le operazioni elencate di seguito:

- Usa Step Functions per orchestrare i flussi di lavoro tra più persone. Servizi AWS
- Usa Amazon Managed Workflows for Apache Airflow (Amazon MWAA) per gestire l'orchestrazione del flusso di lavoro per Apache Airflow.
- Usalo per eseguire e orchestrare le applicazioni Apache Spark. AWS Glue

Puoi migrare i casi d'uso tipici AWS Data Pipeline verso Step Functions o Amazon MWAA. AWS Glue L'opzione scelta dipende dal carico di lavoro corrente su. AWS Data Pipeline Questo argomento spiega come migrare da Step Functions AWS Data Pipeline a Step Functions.

Argomenti

- Migrazione dei carichi di lavoro da AWS Data Pipeline
- Mappatura concettuale tra Step Functions e AWS Data Pipeline
- Progetti di esempio di Step Functions
- Confronto dei prezzi

Migrazione dei carichi di lavoro da AWS Data Pipeline

Step Functions è un servizio di orchestrazione serverless che consente di creare flussi di lavoro per applicazioni aziendali critiche. Con Workflow Studio di Step Functions, puoi creare flussi di lavoro e integrarli con più di 11.000 azioni API tra oltre 250. Servizi AWS Ciò include Servizi AWS Amazon EMR e Amazon DynamoDB. AWS Lambda Puoi anche usare Step Functions per orchestrare le pipeline di elaborazione dei dati, gestire gli errori e lavorare con limiti di limitazione sul sottostante. Servizi AWS Puoi creare flussi di lavoro che elaborano e pubblicano modelli di machine learning, orchestrano microservizi e gestiscono flussi di lavoro di estrazione, trasformazione e caricamento

(ETL). AWS Glue Puoi anche creare flussi di lavoro automatizzati e di lunga durata per applicazioni che richiedono l'interazione umana.

Step Functions è un servizio completamente gestito fornito da AWS. Ciò significa che AWS gestisce al posto tuo attività come la manutenzione dell'infrastruttura, l'applicazione di patch ai lavoratori e la gestione degli aggiornamenti delle versioni del sistema operativo.

Se il tuo caso d'uso soddisfa le seguenti condizioni, ti consigliamo di AWS Data Pipeline migrare da Step Functions:

- Preferisci un servizio di orchestrazione del flusso di lavoro senza server e ad alta disponibilità.
- È necessaria una soluzione che addebiti in base alla granularità dell'esecuzione di una singola attività.
- I tuoi carichi di lavoro prevedono l'orchestrazione di attività per molti altri Servizi AWS, come Amazon EMR, Lambda o DynamoDB. AWS Glue
- Hai bisogno di una soluzione low-code con un visual designer per la creazione di flussi di lavoro. drag-and-drop Questa soluzione non dovrebbe richiedere l'apprendimento di concetti di programmazione complessi e sconosciuti.
- È necessario un servizio che si integri con oltre 250 azioni API Servizi AWS che coprano oltre 11.000 azioni API. Questo servizio deve inoltre integrarsi con servizi e attività personalizzati esterni a. AWS

Mappatura concettuale tra Step Functions e AWS Data Pipeline

AWS Data Pipeline e Step Functions condividono alcuni concetti comuni. Ad esempio, per definire i flussi di lavoro, si utilizza il formato JSON sia in Step Functions che in Step AWS Data Pipeline Functions. In Step Functions, si utilizza Amazon States Language un linguaggio strutturato basato su JSON. Utilizzi Amazon States Language (ASL) per definire i flussi di lavoro e passare dalla rappresentazione testuale a quella visiva del flusso di lavoro. Questo formato basato su JSON aiuta a semplificare l'archiviazione dei flussi di lavoro in uno strumento di controllo del codice sorgente. Inoltre, consente di gestire più versioni dei flussi di lavoro, controllarne l'accesso o automatizzarne l'orchestrazione con metodi CI/CD.

La tabella seguente descrive la mappatura tra i concetti principali utilizzati in entrambi i servizi. La colonna dei concetti di Data pipeline a sinistra elenca i concetti in Step Functions AWS Data Pipeline, mentre la colonna dei concetti di Step Functions a destra elenca i concetti equivalenti in Step Functions.

Concetti relativi alla pipeline di dati	Concetti di Step Functions
Pipeline	Flussi di lavoro
Definizione della pipeline	Amazon States Language(ASL)
Attività	States e Attività
Istanze	Esecuzioni
Tentativi	Cacciatori e retrier
Pianificazione della pipeline	• Esecuzioni con Amazon Scheduler EventBridge • Eventi attivati tramite Pipes EventBridge
Espressioni e funzioni della pipeline	• Funzioni intrinseche • Funzioni Lambda che utilizzano l'integrazione dei servizi

Progetti di esempio di Step Functions

Per un'introduzione a Step Functions, guarda il seguente video:

Guida introduttiva all'orchestrazione AWS Step Functions dei servizi

L'elenco seguente descrive alcuni progetti di esempio che implementano i casi AWS Data Pipeline d'uso più comuni con Step Functions. È possibile utilizzare questi progetti di esempio come riferimento per la migrazione AWS Data Pipeline da Step Functions. Puoi anche usarli come base per creare flussi di lavoro personalizzati e integrarli con quelli supportati in Servizi AWS base al tuo caso d'uso.

- Gestisci un lavoro in Amazon EMR
- Esegui un processo di elaborazione dati su Amazon EMR Serverless
- Esecuzione di processi Hive/Pig/Hadoop
- Interroga set di dati di grandi dimensioni (Amazon Athena, Amazon S3 e Amazon SNS AWS Glue)
- Esegui flussi di lavoro ETL/ELT con Amazon Redshift

- AWS Glue Orchestrazione dei crawler
- Esegui uno script di shell con Step Functions

Per ulteriori informazioni su Step Functions, consulta i seguenti argomenti e risorse:

- Tutorial per Step Functions
- Progetti di esempio per Step Functions
- Il AWS Step Functions workshop

Confronto dei prezzi

AWS Data Pipeline ha un prezzo in base al numero di condotte e al loro livello di utilizzo. Le attività eseguite più di una volta al giorno (alta frequenza) hanno un prezzo di 1 USD al mese per attività. Le attività eseguite una volta al giorno o meno (bassa frequenza) hanno un prezzo di 0,60 USD al mese per attività. Le pipeline inattive hanno un prezzo di 1 USD per pipeline. Per ulteriori informazioni sui prezzi, consulta la pagina dei prezzi. AWS Data Pipeline

Step Functions ha due tipi di flussi di lavoro: Standard ed Express. Ogni tipo di flusso di lavoro ha un modello di prezzo diverso. Questo confronto si basa sul flusso di lavoro Standard poiché corrisponde meglio ai casi d'uso comuni di AWS Data Pipeline. I flussi di lavoro standard hanno un prezzo di 0,025 USD per 1000 transizioni di stato. Non è previsto alcun costo per le macchine a stati inattive; si paga solo per ciò che si utilizza. Per ulteriori informazioni sui prezzi, consulta la pagina AWS Step Functions dei prezzi.

Risoluzione dei problemi

In caso di difficoltà nell'utilizzo di Step Functions, utilizza le seguenti risorse per la risoluzione dei problemi.

Argomenti

- Risoluzione dei problemi generali
- Risoluzione dei problemi di integrazione dei servizi
- Attività di risoluzione dei problemi
- Risoluzione dei problemi dei flussi di lavoro Express

Risoluzione dei problemi generali

Non riesco a creare una macchina a stati.

Il ruolo IAM associato alla macchina a stati potrebbe non disporre di autorizzazioni sufficienti. Controlla le autorizzazioni del ruolo IAM, incluse le attività di integrazione dei AWS servizi, X-Ray e CloudWatch la registrazione. Sono necessarie autorizzazioni aggiuntive per gli stati `.sync` delle attività.

Non riesco a usare a JsonPath per fare riferimento all'output dell'attività precedente.

Per aJsonPath, una chiave JSON deve terminare con`.$`. Ciò significa che a JsonPath può essere utilizzato solo in una coppia chiave-valore. Se si desidera utilizzare JsonPath altri luoghi, ad esempio un array, è possibile utilizzare funzioni intrinseche. Ad esempio, potresti usare qualcosa di simile al seguente:

Uscita dell'attività A:

```
{
    "sample": "test"
}
```

Compito B:

```
{
    "JsonPathSample.$": "$.sample"
}
```

> **Tip**
>
> Usa il simulatore di flusso di dati nella console Step Functions per testare la sintassi dei percorsi JSON, per comprendere meglio come i dati vengono manipolati all'interno di uno stato e per vedere come i dati vengono passati tra gli stati.

C'è stato un ritardo nelle transizioni di stato.

Per i flussi di lavoro standard, esiste un limite al numero di transizioni di stato. Quando si supera il limite di transizione di stato, Step Functions ritarda le transizioni di stato fino al riempimento del bucket per la quota. La limitazione del limite di transizione dello stato può essere monitorata esaminando la `ExecutionThrottled` metrica nella Parametri di esecuzione sezione della CloudWatch pagina Metriche.

Quando avvio nuove esecuzioni di Standard Workflow, queste falliscono con l'**`ExecutionLimitExceeded`**errore.

Step Functions ha un limite di 1.000.000 di esecuzioni aperte per ciascuna Account AWS di esse. Regione AWS Se si supera questo limite, Step Functions genera un `ExecutionLimitExceeded` errore. Questo limite non si applica ai flussi di lavoro Express. Puoi utilizzare la seguente CloudWatchmetrica matematica nella Amazon CloudWatch User Guide per approssimare il numero di esecuzioni aperte:. `ExecutionsStarted - (ExecutionsSucceeded + ExecutionsTimedOut + ExecutionsFailed + ExecutionsAborted)`

Un errore su un ramo in uno stato parallelo causa il fallimento dell'intera esecuzione.

Si tratta di un comportamento previsto. Per evitare errori quando si utilizza uno stato parallelo, configura Step Functions per rilevare gli errori generati da ogni ramo.

Risoluzione dei problemi di integrazione dei servizi

Il mio lavoro è completo nel servizio a valle, ma in Step Functions lo stato dell'attività rimane «In corso» o il suo completamento è ritardato.

Per i modelli di integrazione dei `.sync` servizi, Step Functions utilizza EventBridge regole, API downstream o una combinazione di entrambe per rilevare lo stato del processo a valle. Per alcuni servizi, Step Functions non crea EventBridge regole da monitorare. Ad esempio, per l'integrazione del AWS Glue servizio, invece di utilizzare EventBridge le regole, Step Functions effettua una `glue:GetJobRun` chiamata. A causa della frequenza delle chiamate API, esiste una differenza tra il completamento dell'attività a valle e il tempo di completamento dell'attività di Step Functions. Step Functions richiede le autorizzazioni IAM per gestire le EventBridge regole ed effettuare chiamate al servizio downstream. Per ulteriori dettagli su come le autorizzazioni insufficienti sul ruolo di esecuzione possono influire sul completamento delle attività, vedereAutorizzazioni aggiuntive per le attività che utilizzano il pattern Run a Job.

Voglio restituire un output JSON dall'esecuzione di una macchina a stati annidati.

Esistono due integrazioni sincrone del servizio Step Functions per Step Functions: `startExecution.sync` e. `startExecution.sync:2` Entrambi attendono il completamento della macchina a stati annidati, ma restituiscono `Output` formati diversi. Puoi usarlo `startExecution.sync:2` per restituire un output JSON sotto`Output`.

Non riesco a richiamare una funzione Lambda da un altro account.

Accesso alla funzione Lambda con supporto multiaccount

Se l'accesso alle AWS risorse tra account è disponibile nella tua regione, utilizza il seguente metodo per richiamare una funzione Lambda da un altro account.

Per richiamare una risorsa multiaccount nei tuoi flussi di lavoro, procedi come segue:

1. Crea un ruolo IAM nell'account di destinazione che contiene la risorsa. Questo ruolo concede all'account di origine, contenente la macchina a stati, le autorizzazioni per accedere alle risorse dell'account di destinazione.
2. Nella definizione dello `Task` stato, specifica il ruolo IAM di destinazione che deve essere assunto dalla macchina a stati prima di richiamare la risorsa multiaccount.

3. Modifica la politica di fiducia nel ruolo IAM di destinazione per consentire all'account di origine di assumere temporaneamente questo ruolo. La policy di affidabilità deve includere l'Amazon Resource Name (ARN) della macchina a stati definita nell'account di origine. Inoltre, definisci le autorizzazioni appropriate nel ruolo IAM di destinazione per chiamare la AWS risorsa.
4. Aggiorna il ruolo di esecuzione dell'account di origine per includere l'autorizzazione richiesta per assumere il ruolo IAM di destinazione.

Per un esempio, consulta Tutorial: Accesso alle risorse di più account AWS.

> **Note**
>
> Puoi configurare la tua macchina a stati in modo che assuma un ruolo IAM per accedere a risorse da più Account AWS risorse. Tuttavia, una macchina a stati può assumere un solo ruolo IAM alla volta.

Per un esempio di definizione di `Task` stato che specifica una risorsa multiaccount, vedere. Esempi di campi relativi alle credenziali dello stato dell'attività

Accesso alla funzione Lambda senza supporto tra account

Se l'accesso alle AWS risorse tra account non è disponibile nella tua regione, utilizza il seguente metodo per richiamare una funzione Lambda da un altro account.

Nel `Resource` campo dello `Task` stato, usa `arn:aws:states:::lambda:invoke` e passa i parametri `FunctionArn` in. Il ruolo IAM associato alla macchina a stati deve avere le autorizzazioni giuste per richiamare le funzioni Lambda tra account:. `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
      }
   }
```

```
}
```

Non riesco a vedere i task token trasmessi dagli `.waitForTaskToken` stati.

Nel `Parameters` campo dello `Task` stato, devi passare un task token. Ad esempio, potresti usare qualcosa di simile al codice seguente.

```
{
   "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**
>
> Puoi provare a utilizzarlo `.waitForTaskToken` con qualsiasi azione API. Tuttavia, alcune API non dispongono di parametri adeguati.

Attività di risoluzione dei problemi

L'esecuzione della mia macchina a stati è bloccata in uno stato di attività.

Lo stato di un'attività non viene avviato finché non si esegue il sondaggio di un token dell'attività utilizzando l'azione GetActivityTaskAPI. Come procedura consigliata, aggiungi un timeout a livello di attività per evitare un'esecuzione bloccata. Per ulteriori informazioni, consulta Utilizza i timeout per evitare esecuzioni bloccate.

Se la tua macchina statale è bloccata nell'ActivityScheduledevento, significa che la tua flotta di lavoratori in attività presenta problemi o è sottodimensionata. È necessario monitorare la `ActivityScheduleTime`CloudWatchmetrica e impostare una sveglia quando tale tempo aumenta. Tuttavia, per interrompere eventuali esecuzioni di macchine a stati bloccati in cui lo `Activity` stato non passa allo `ActivityStarted` stato, definisci un timeout a livello di macchina a stati. Per fare ciò, specifica un `TimeoutSeconds` campo all'inizio della definizione della macchina a stati, all'esterno del `States` campo.

Il mio assistente in attività scade in attesa di un task token.

I lavoratori utilizzano l'azione GetActivityTaskAPI per recuperare un'attività con l'attività specificata ARN che è pianificata per l'esecuzione da parte di una macchina a stati in esecuzione. `GetActivityTask`avvia un lungo sondaggio, in modo che il servizio mantenga aperta la connessione HTTP e risponda non appena un'attività diventa disponibile. Il tempo massimo in cui il servizio trattiene la richiesta prima di rispondere è di 60 secondi. Se nessuna attività è disponibile entro 60 secondi, il sondaggio restituisce una stringa `taskToken` con una stringa nulla. Per evitare questo timeout, configura un socket lato client con un timeout di almeno 65 secondi nell'AWSSDK o nel client che stai utilizzando per effettuare la chiamata API.

Risoluzione dei problemi dei flussi di lavoro Express

La mia applicazione si interrompe prima di ricevere una risposta da una chiamata **`StartSyncExecution`** API.

Configura un timeout del socket lato client nell'AWSSDK o nel client che usi per effettuare la chiamata API. Per ricevere una risposta, il timeout deve avere un valore superiore alla durata delle esecuzioni di Express Workflow.

Non riesco a visualizzare la cronologia delle esecuzioni per risolvere gli errori di Express Workflow.

Express Workflow non registra la cronologia delle esecuzioni inAWS Step Functions. È invece necessario attivare la CloudWatch registrazione. Una volta attivata la registrazione, puoi utilizzare le query di CloudWatch Logs Insights per esaminare le esecuzioni di Express Workflow. È inoltre possibile visualizzare la cronologia delle esecuzioni di Express Workflow nella console Step Functions se si sceglie il pulsante Abilita nella scheda Esecuzioni. Per ulteriori informazioni, consulta Visualizzazione e debug delle esecuzioni sulla console Step Functions.

Per elencare le esecuzioni in base alla durata:

```
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
```

Per elencare le esecuzioni non riuscite e annullate:

```
fields ispresent(execution_arn) as isRes | filter type in ["ExecutionFailed",
 "ExecutionAborted", "ExecutionTimedOut"]
```

Informazioni correlate

La tabella seguente elenca le risorse correlate che possono essere utili durante l'utilizzo di questo servizio.

Risorsa	Descrizione
AWS Step Functions Documentazione di riferimento delle API	Descrizione di operazioni, parametri e tipi di dati API e un elenco di errori generati dal servizio.
AWS Step FunctionsRiferimento alla riga di comando	Descrizioni dei comandi di AWS CLI che possono essere utilizzati con AWS Step Functions.
Informazioni sul prodotto per Step Functions	La pagina Web principale per informazioni su Step Functions.
Forum di discussione	Un forum basato sulla community per gli sviluppatori per discutere di domande tecniche relative a Step Functions e ad altri servizi. AWS
AWS SupportInformazioni	La pagina Web principale per informazioni suAWS Support, a, un canale di ne-on-one supporto a risposta rapida per aiutarti a creare ed eseguire applicazioni sui AWS servizi di infrastruttura.

Lancio recente di funzionalità

La tabella seguente elenca le regioni in cui sono disponibili le nuove funzionalità di Step Functions.

Data di lancio	Nome della funzionalità	Regioni disponibili
26 novembre 2023	Richiama gli endpoint HTTPS pubblici e testa i singoli stati	• Stati Uniti orientali (Virginia settentrionale) - us-east-1 • Stati Uniti occidentali (Oregon) - us-west-2 • Stati Uniti orientali (Ohio) - us-east-2 • Europa (Irlanda) - eu-west-1 • UE (Francoforte) - eu-central-1 • Europa (Stoccolma) - eu-nord-1 • Asia Pacifico (Sydney) - ap-southeast-2 • Asia Pacifico (Tokyo) - ap-northeast-1 • Asia Pacifico (Singapore) - ap-southeast-1
15 novembre 2023	Redriveesecuzioni	Per un elenco completo delle aree Regioni AWS in cui questa funzionalità è disponibile, consulta le opzioni nell'elenco a discesa Regione nella pagina intitolata Servizi AWS per regione.
12 ottobre 2023	Integrazione ottimizzata per Amazon EMR Serverless	Per un elenco completo delle aree Regioni AWS in cui questa funzionalità è

Data di lancio	Nome della funzionalità	Regioni disponibili
		disponibile, consulta le opzioni nell'elenco a discesa Regione nella pagina intitolata Servizi AWSper regione.
07 settembre 2023	Gestione degli errori migliorata	Per un elenco completo delle aree Regioni AWS in cui questa funzionalità è disponibile, consulta le opzioni nell'elenco a discesa Regione nella pagina intitolata Servizi AWSper regione.
31 agosto 2023	Miglioramenti di Workflow Studio per un'esperienza di creazione semplificata	Per un elenco completo delle aree Regioni AWS in cui questa funzionalità è disponibile, consulta le opzioni nell'elenco a discesa Regione nella pagina intitolata per regione. Servizi AWS
22 giugno 2023	Versioni e alias	Per un elenco completo delle aree Regioni AWS in cui questa funzionalità è disponibile, consulta le opzioni nell'elenco a discesa Regione nella pagina intitolata Servizi AWS per regione.
16 giugno 2023	Nuove integrazioni SDK AWS	Per un elenco completo delle aree Regioni AWS in cui questa funzionalità è disponibile, consulta le opzioni nell'elenco a discesa Regione nella pagina intitolata Servizi AWS per regione.

Data di lancio	Nome della funzionalità	Regioni disponibili
01 dicembre 2022	Orchestra flussi di lavoro paralleli su larga scala per l'elaborazione dei dati con Distributed Map state	Per un elenco completo delle aree Regioni AWS in cui questa funzionalità è disponibile, consulta le opzioni nell'elenco a discesa Regione nella pagina intitolata per regione. Servizi AWS

Cronologia dei documenti

Questa sezione elenca le principali modifiche alla Guida per gli AWS Step Functions sviluppatori.

Modifica	Descrizione	Data della modifica
Nuova caratteristica	Step Functions aggiunge un'integrazione ottimizzata per AWS Elemental MediaConvert AWS Elemental MediaConvert offre la transcodifica di file audio e video di livello broadcast, che i clienti possono automatizzare con codice per adattarlo ai propri flussi di lavoro multimediali. Grazie all'integrazione ottimizzata per AWS Step Functions in MediaConvert, è ora possibile orchestrare utilizzando lo strumento visivo a basso codice Workflow Studio. Per ulteriori informazioni, consulta la documentazione relativa a Manage AWS Elemental MediaConvert with Step Functions.	12 aprile 2024
Aggiornamenti	AWS aggiornamenti delle politiche gestite: aggiornamento a una politica esistente Sono state aggiunte informazioni sulle nuove autorizzazioni di sola lettura per tag, mappe distribuite, versioni e alias. Per ulteriori informazioni, consulta AWS politiche gestite per AWS Step Functions.	02 aprile 2024
Aggiornamenti	Step Functions aggiunge il supporto per le metriche di Open Workflow Con le metriche aperte del flusso di lavoro, ora hai una visibilità a livello di account sul numero di flussi di lavoro standard in corso e sul limite del flusso di lavoro aperto. Puoi gestire i carichi di lavoro in tutti i flussi di lavoro, indipendentemente da come sono stati avviati, per garantire operazioni fluide del flusso di lavoro. Puoi impostare CloudWatch allarmi per monitorare i flussi di	29 febbraio 2024

Modifica	Descrizione	Data della modifica
	lavoro e ricevere in modo proattivo avvisi man mano che ti avvicini ai limiti. Una volta ricevuti gli avvisi, puoi gestire efficacemente i flussi di lavoro intraprendendo azioni come l'interruzione di flussi di lavoro specifici o la richiesta di un aumento del limite. Le metriche aperte del flusso di lavoro sono disponibili per l'uso nei flussi di lavoro standard senza che sia CloudWatch richiesta alcuna configurazione aggiuntiva. Per ulteriori informazioni, consulta Parametri di esecuzione.	
Aggiornamenti	Aggiunte e aggiornamenti per l'integrazione dei servizi. Per l'elenco delle integrazioni AWS SDK nuove e aggiornate, consulta. Registro delle modifiche per le integrazioni AWS SDK supportate Per l'elenco completo dei servizi, consulta. Integrazioni di servizi AWS SDK supportate	18 gennaio 2024
Nuova caratteristica	Usa Workflow Studio Application Composer per creare flussi di lavoro serverless utilizzando modelli. AWS CloudFormation Per ulteriori informazioni, consulta Utilizzo di Workflow Studio in Application Composer.	27 novembre 2023
Nuova caratteristica	Step Functionsora ti consente di richiamare direttamente gli endpoint HTTPS pubblici e di testare i singoli stati utilizzando una nuova API Test State. Per ulteriori informazioni, consultare: • Chiama API di terze parti • Utilizzo TestState dell'API per testare uno stato	26 novembre 2023

Modifica	Descrizione	Data della modifica
Nuova caratteristica	Step Functions si integra ora con Amazon Bedrock Per ulteriori informazioni, consulta i seguenti argomenti: • Chiama Amazon Bedrock con Step Functions • Autorizzazioni IAM per Amazon Bedrock • Esegui il concatenamento dei prompt AI con Amazon Bedrock • Utilizzo AWS Step Functions con altri servizi	26 novembre 2023
Nuova caratteristica	Step Functions ora consente redrive di eseguire flussi di lavoro di tipo Standard dal punto di errore. Per ulteriori informazioni, consulta Redrivingesecuzioni e Redriving Mappa corre.	15 novembre 2023
Aggiornamento solo della documentazione	È stato pubblicato un nuovo argomento che spiega come eseguire macchine a stati in base a una pianificazione utilizzando. Amazon EventBridge Scheduler Per ulteriori informazioni, consulta Utilizzo di Amazon EventBridge Scheduler con AWS Step Functions.	16 ottobre 2023
Nuova caratteristica	Step Functions si integra ora con Amazon EMR Serverless Per ulteriori informazioni, consulta i seguenti argomenti: • Chiama Amazon EMR Serverless con Step Functions • Esegui un EMR Serverless lavoro • Integrazioni ottimizzate per Step Functions • Utilizzo AWS Step Functions con altri servizi	12 ottobre 2023
Aggiornamento solo della documentazione	Sono state aggiunte informazioni sull'esecuzione di macchine a stati in base a una pianificazione utilizzando. Amazon EventBridge Scheduler Per ulteriori informazioni, consulta Usare Scheduler EventBridge .	5 ottobre 2023

Modifica	Descrizione	Data della modifica
Aggiornamento	Gli argomenti relativi allo stato della mappa distribuita sono stati riorganizzati e aggiornati per motivi di chiarezza e brevità e per creare una mappa di viaggio chiara per i nuovi utenti. Per ulteriori informazioni, consulta Utilizzo dello stato della mappa in modalità distribuita per orchestrare carichi di lavoro paralleli su larga scala.	6 ottobre 2023
Correzioni	Esempi di codice corretti in un tutorial per l'uso AWS CDK della v2. Per ulteriori informazioni, consulta Creazione di una macchina a Lambda stati per Step Functions l'utilizzo AWS CDK.	19 settembre 2023
Aggiornamento	Sono state aggiunte informazioni sulle funzionalità avanzate di gestione degli errori introdotte da Step Functions per identificare chiaramente gli errori e implementare nuovi tentativi con maggiore controllo. Per ulteriori informazioni, consulta Fail e Riprovare dopo un errore.	07 settembre 2023
Aggiornamento	Step Functions ha aggiunto miglioramenti a Workflow Studio per semplificare l'esperienza di creazione dei flussi di lavoro. Per ulteriori informazioni, consulta AWS Step Functions Workflow Studio.	31 agosto 2023
Aggiornamento solo della documentazione	Sono state aggiunte informazioni sul doppio del conteggio effettivo delle metriche riportato per la metrica. `ExecutionsStarted` Per ulteriori informazioni, consulta Metriche che riportano un conteggio.	25 luglio 2023
Aggiornamento solo della documentazione	Step Functions ha aggiunto due nuovi progetti di esempio che dimostrano i seguenti casi d'uso comuni per lo stato della mappa distribuita: • Elaborazione di un file CSV • Elaborazione dei dati in un bucket Amazon S3	17 luglio 2023

Modifica	Descrizione	Data della modifica
Aggiornamento solo della documentazione	Pubblicato un nuovo argomento sulla distribuzione di macchine a stati utilizzando Terraform. Per ulteriori informazioni, consulta Distribuzione di macchine a stati utilizzando Terraform.	5 luglio 2023
Aggiornamento solo della documentazione	Sono state aggiornate le seguenti procedure per far corrispondere le modifiche all' EventBridge interfaccia Amazon. • Instradamento di un evento Step Functions a EventBridge • Avvio dell'esecuzione di una macchina a stati in risposta agli eventi di Amazon S3	26 giugno 2023
Nuova caratteristica	Step Functions ora offre la possibilità di creare più versioni e alias di macchine a stati per una maggiore resilienza durante l'implementazione di flussi di lavoro serverless. Per ulteriori informazioni, consulta Gestisci le distribuzioni continue con versioni e alias.	22 giugno 2023
Aggiornamento solo della documentazione	È stata migliorata la descrizione `TimeoutSeconds` e `HeartbeatSeconds` i campi per descrivere in che modo sono diversi l'uno dall'altro. Per ulteriori informazioni, consulta Campi relativi allo stato delle attività.	22 giugno 2023
Aggiornamento solo della documentazione	È stata pubblicata una nuova sezione che descrive come appiattire un array di array generalmente restituiti come risultato degli stati Parallel e Map. Per ulteriori informazioni, consulta Appiattimento di una serie di array.	20 giugno 2023
Aggiornamento	Step Functions ha ampliato il supporto per le integrazioni AWS SDK aggiungendo sette Servizi AWS e 468 nuove azioni API. Per ulteriori informazioni, consulta Integrazioni di servizi AWS SDK supportate e Registro delle modifiche per le integrazioni AWS SDK supportate.	16 giugno 2023

Modifica	Descrizione	Data della modifica
Aggiornamento solo della documentazione	È stato pubblicato un nuovo argomento che elenca le Regioni AWS funzionalità Step Functions lanciate di recente. Per ulteriori informazioni, consulta Lancio recente di funzionalità.	16 giugno 2023
Aggiornamento solo della documentazione	Step Functions ora include una sezione su Notifiche all'utente AWS, an Servizio AWS che funge da posizione centrale per le AWS notifiche in AWS Management Console. Per ulteriori informazioni, consulta Utilizzo Notifiche all'utente AWS con Step Functions.	4 maggio 2023
Aggiornamento solo della documentazione	È stata aggiunta una nuova sezione che spiega le autorizza zioni necessarie per scrivere i risultati dell'esecuzione del flusso di lavoro secondario su un bucket Amazon S3 crittografato con una chiave. AWS Key Managemen t Service (AWS KMS) Per ulteriori informazioni, consulta Autorizzazioni IAM per bucket Amazon S3 AWS KMS key crittografato.	25 aprile 2023
Aggiornamento solo della documentazione	È stato aggiunto un nuovo argomento che spiega la funzionalità di simulazione del flusso di dati. Per ulteriori informazioni, consulta Simulatore di flusso di dati.	14 aprile 2023
Aggiornamento delle quote	Sono state aggiunte informazioni sulla quota predefinita di 1000 per Map Run aperte in ogni account. Per ulteriori informazioni, consulta Quote relative agli account.	05 aprile 2023
Aggiornamento solo della documentazione	È stato aggiunto un argomento che descrive quando migrare i AWS Data Pipeline carichi di lavoro a Step Functions. Questo argomento fornisce anche un elenco di esempi che spiegano come eseguire la migrazione. Per ulteriori informazioni, consulta Migrazione dei carichi di lavoro da Step AWS Data Pipeline Functions.	30 marzo 2023

Modifica	Descrizione	Data della modifica
Aggiornamento solo della documentazione	È stata aggiunta una nota sull'indisponibilità del tracciamento a raggi X per lo stato della mappa distribuita. Per ulteriori informazioni, consulta AWS X-Ray e Step Functions.	21 marzo 2023
Aggiornamento solo della documentazione	Sono state aggiunte informazioni su come Step Functions gestisce l'autorizzazione basata su tag. Per ulteriori informazioni, consulta Inserimento di tag in Step Functions e Policy basate su tag.	15 marzo 2023
Aggiornamento solo della documentazione	Sono state aggiunte informazioni su come Step Functions analizza i file CSV utilizzati come input nello stato della mappa distribuita. Per ulteriori informazioni, consulta File CSV in un bucket Amazon S3.	14 marzo 2023
Aggiornamento solo della documentazione	Sono state aggiunte informazioni su come Step Functions gestisce le chiamate tra account per il pattern Run a Job (.sync). Per ulteriori informazioni, consulta Run a Job (.sync).	1 marzo 2023
Aggiornamento solo della documentazione	Riduci il periodo di conservazione della cronologia delle esecuzioni completate del flusso di lavoro da 90 giorni a 30 giorni. Per ulteriori informazioni sulla regolazione del periodo di conservazione, consulta Garanzie di esecuzione e. Quote relative alle esecuzioni di macchine a stati	21 febbraio 2023
Aggiornamento	Step Functions ha ampliato il supporto per le integrazioni AWS SDK aggiungendo 35 AWS servizi e 1100 nuove azioni API. Per ulteriori informazioni, consulta Integrazioni di servizi AWS SDK supportate e Registro delle modifiche per le integrazioni AWS SDK supportate.	17 febbraio 2023

Modifica	Descrizione	Data della modifica
Aggiornamento solo della documentazione	Ha pubblicato una serie di tutorial Getting Started che illustra il processo di creazione di un flusso di lavoro per l'applicazione di carte di credito utilizzando Step Functions. Per ulteriori informazioni, consulta Nozioni di base su AWS Step Functions.	30 dicembre 2022
Nuova caratteristica	Step Functions aggiunge il supporto per orchestrare flussi di lavoro paralleli su larga scala per l'elaborazione dei dati utilizzando una nuova modalità distribuita per lo stato. `Map` Per ulteriori informazioni, consulta Utilizzo dello stato della mappa in modalità distribuita per orchestrare carichi di lavoro paralleli su larga scala.	01 dicembre 2022
Nuova caratteristica	Step Functions ora supporta l'accesso a AWS risorse tra account configurate in altri account. Per ulteriori informazioni, consulta la pagina • Accesso a risorse Account AWS in altri flussi di lavoro • Tutorial: Accesso alle risorse di più account AWS • Task state	18 novembre 2022
Aggiornamento	Step Functions offre ora una nuova esperienza di console per la visualizzazione e il debug delle esecuzioni dei flussi di lavoro Express. Per ulteriori informazioni, consulta la pagina: • Esecuzioni Standard ed Express Workflow nella console • Visualizzazione e debug delle esecuzioni sulla console Step Functions	18 ottobre 2022

Modifica	Descrizione	Data della modifica
Aggiornamento	È stato aggiunto il supporto per specificare facoltativamente il `ExecutionRoleArn` parametro durante l'utilizzo delle `addStep.sync` API `addStep` e per l'integrazione dei servizi ottimizzata di Amazon EMR. Per ulteriori informazioni, consulta Chiama Amazon EMR con Step Functions.	20 settembre 2022
Aggiornamento solo della documentazione	È stato aggiunto un nuovo argomento che fornisce consigli sull'ottimizzazione dei costi durante la creazione di flussi di lavoro serverless utilizzando Step Functions. Per ulteriori informazioni, consulta Ottimizzazione dei costi con Express Workflows.	15 settembre 2022

Modifica	Descrizione	Data della modifica
Aggiornamento	Step Functions aggiunge il supporto per 14 nuove funzioni intrinseche per l'esecuzione di attività di elaborazione dei dati, come manipolazioni di array, codifica e decodifica dei dati, calcoli hash, manipolazione dei dati JSON, operazioni di funzioni matematiche e generazione di identificatori univoci. Aggiornamento solo della documentazione: Ha raggruppato tutte le funzioni intrinseche esistenti e quelle appena introdotte nelle seguenti categorie in base al tipo di attività di elaborazione dei dati che consentono di eseguire: • Elementi intrinseci per gli array • Intrinseci per la codifica e la decodifica dei dati • Intrinseco per il calcolo dell'hash • Caratteristiche intrinseche per la manipolazione dei dati JSON • Elementi intrinseci per le operazioni matematiche • Intrinseco per il funzionamento con stringhe • Intrinseco per la generazione di identificatori univoci • Intrinseco per operazioni generiche Per ulteriori informazioni, consulta Funzioni intrinseche.	31 agosto 2022
Aggiornamento	Step Functions ha ampliato il supporto per le integrazioni AWS SDK aggiungendo altri tre AWS servizi: AWS Billing ConductorAmazon GameSparks, e. Amazon Pinpoint SMS and Voice V2 Per ulteriori informazioni, consulta Registro delle modifiche per le integrazioni AWS SDK supportate.	26 luglio 2022

Modifica	Descrizione	Data della modifica
Aggiornamento solo della documentazione	È stato aggiunto un nuovo argomento per includere un riepilogo di tutti gli aggiornamenti apportati alle integrazioni AWS SDK supportate da Step Functions. Per ulteriori informazioni, consulta Registro delle modifiche per le integrazioni AWS SDK supportate	26 luglio 2022
Aggiornamento solo della documentazione	AWS Step Functions La Guida per gli sviluppatori ora include dettagli sulle metriche di esecuzione emesse specificamente per Express Workflows. Per ulteriori informazioni, consulta Metriche di esecuzione per Express Workflows.	09 giugno 2022

Modifica	Descrizione	Data della modifica
Aggiornamento	Miglioramenti alla console Step Functions La console ora presenta una pagina dei dettagli di esecuzione riprogettata che include i seguenti miglioramenti: • Capacità di identificare a colpo d'occhio il motivo di un'esecuzione non riuscita. • Due nuove modalità di visualizzazione per la vostra macchina a stati: visualizzazione tabella e visualizzazione eventi. Queste visualizzazioni offrono anche la possibilità di applicare filtri per visualizzare solo le informazioni di interesse. Inoltre, è possibile ordinare i contenuti della visualizzazione degli eventi in base ai timestamp dell'evento. • Passa tra le diverse iterazioni di `Map` stato nella modalità di visualizzazione Grafico utilizzando un elenco a discesa o nella visualizzazione ad albero della modalità di visualizzazione Tabella per gli stati. `Map` • Visualizza informazioni dettagliate su ogni stato del flusso di lavoro, incluso il percorso completo di trasferimento dei dati di input e output e i tentativi di riprovare per gli stati o. `Task` `Parallel` • Vari miglioramenti, tra cui la possibilità di copiare l'esecuzione della macchina a stati Amazon Resource Name, visualizzare il numero totale di transizioni della macchina a stati ed esportare i dettagli di esecuzione in formato JSON. Aggiornamenti solo relativi alla documentazione È stato aggiunto un nuovo argomento per spiegare i vari tipi di informazioni visualizzate nella pagina Dettagli	09 maggio 2022

Modifica	Descrizione	Data della modifica
	di esecuzione. Inoltre, è stato aggiunto un tutorial per mostrare come esaminare queste informazioni. Per ulteriori informazioni, consultare: • Visualizzazione e debug delle esecuzioni sulla console Step Functions • Tutorial: Esame delle esecuzioni delle macchine a stati utilizzando la console Step Functions	
Aggiornamento	Step Functions offre ora una soluzione alternativa per evitare il confuso problema della sicurezza secondaria, che si verifica quando un'entità (un servizio o un account) viene costretta da un'altra entità a eseguire un'azione. Per ulteriori informazioni, consultare: • Evita problemi secondari confusi tra diversi servizi	02 maggio 2022

Modifica	Descrizione	Data della modifica

Aggiornamento	• Step Functions ha ampliato il supporto per le integrazioni AWS SDK aggiungendo altri AWS 21 servizi. Per ulteriori informazioni, consulta Integrazioni di servizi AWS SDK supportate.
	• Aggiornamenti solo relativi alla documentazione:
	  • È stato aggiunto un elenco di tutti i prefissi di eccezione presenti nelle eccezioni generate quando si esegue erroneamente un'integrazione del AWS servizio SDK con Step Functions. Per ulteriori informazioni, consulta Integrazioni di servizi AWS SDK supportate.
	  • È stato aggiunto un elenco di tutte le azioni API non supportate per le integrazioni SDK supportate. AWS Per ulteriori informazioni, consulta Azioni API non supportate per i servizi supportati.
	  • È stato aggiunto un elenco di tutte le integrazioni AWS SDK supportate che ora sono obsolete. Per ulteriori informazioni, consulta Integrazioni di servizi SDK obsolete AWS.
		19 aprile 2022

Nuova caratteristica	Step Functions Local ora supporta l'integrazione AWS SDK e la simulazione delle integrazioni di servizi. Per ulteriori informazioni, consultare: • Utilizzo di integrazioni di servizi simulati	28 gennaio 2022
Nuova caratteristica	AWS Step Functions ora supporta la creazione di un'API REST di Amazon API Gateway con macchina a stati express sincrona come integrazione di backend utilizzan do. AWS Cloud Development Kit (AWS CDK) Per ulteriori informazioni, consultare: • Creazione di un'API Gateway REST con Synchronous Express State Machine utilizzando AWS CDK	10 dicembre 2021

Modifica	Descrizione	Data della modifica
Aggiornamento	Step Functions ha aggiunto tre nuovi progetti di esempio che dimostrano l'integrazione di Step Functions e la console aggiornata di Amazon Athena. Per ulteriori informazioni, consultare: • Esegui più query (Amazon Athena, Amazon SNS) • Interroga set di dati di grandi dimensioni (Amazon Athena, Amazon S3 e Amazon SNS AWS Glue) • Mantieni aggiornati i dati (Amazon Athena, Amazon S3,) AWS Glue	22 novembre 2021
Nuova caratteristica	Step Functions ha aggiunto il supporto degli endpoint Amazon VPC per i flussi di lavoro Synchronous Express. Per ulteriori informazioni, consultare: • Endpoint Amazon VPC per Step Functions	15 novembre 2021
Aggiornamento	AWS Step Functions ha aggiunto tre nuovi progetti di esempio che dimostrano come utilizzare l' AWS Batch integrazione Step Functions. Per ulteriori informazioni, consultare: • Trova un AWS Batch lavoro • AWS Batch con Lambda • Utilizzo Step Functions e AWS Batch con gestione degli errori	14 ottobre 2021

Modifica	Descrizione	Data della modifica
Nuova caratteristica	AWS Step Functions ha aggiunto integrazioni AWS SDK, che consentono di utilizzare le azioni API per tutti gli oltre duecento servizi. AWS Per ulteriori informazioni, consultare: • AWS Integrazioni di servizi SDK • Raccogli informazioni sui bucket Amazon S3 utilizzando AWS le integrazioni dei servizi SDK	30 settembre 2021
Nuova caratteristica	AWS Step Functions ha aggiunto un designer di flussi di lavoro visivi, Workflow Studio. AWS Step Functions Per ulteriori informazioni, consultare: • AWS Step Functions Workflow Studio • Impara a usare AWS Step Functions Workflow Studio	17 giugno 2021
Aggiornamento	AWS Step Functions ha aggiunto quattro nuove API, `StartBuildBatch` `StopBuildBatch` , `RetryBuildBatch` `eDeleteBuildBatch` , all' CodeBuild integrazione. Per ulteriori informazioni, consultare: • Chiama AWS CodeBuild con Step Functions	4 giugno 2021
Nuova caratteristica	AWS Step Functions ora si integra con Amazon EventBridge. Per ulteriori informazioni, consultare: • Chiama EventBridge con Step Functions • Politiche IAM per Step Functions e Politiche IAM per Amazon EventBridge • Un progetto di esempio che mostra come Invia un evento personalizzato a EventBridge	14 maggio 2021

Modifica	Descrizione	Data della modifica
Aggiornamento	AWS Step Functions ha aggiunto un nuovo progetto di esempio che mostra come utilizzare Step Functions e l'Amazon Redshift Data API per eseguire un flusso di lavoro ETL/ELT. Per ulteriori informazioni, consultare: • Esegui flussi di lavoro ETL/ELT utilizzando Amazon Redshift (Lambda, Amazon Redshift Data API)	16 aprile 2021
Nuova caratteristica	AWS Step Functions ha un nuovo simulatore di flusso di dati nella console. Per ulteriori informazioni, consultare: • Console Step Functions	8 aprile 2021
Nuova caratteristica	AWS Step Functions ora si integra con Amazon EMR su EKS. Per ulteriori informazioni, consultare: • Chiama Amazon EMR su EKS con AWS Step Functions	29 marzo 2021
Aggiornamento	Il supporto YAML per le definizioni delle macchine a stati è stato aggiunto a e. AWS Toolkit for Visual Studio Code AWS CloudFormation Per ulteriori informazioni, consultare: • Supporto per il formato di definizione • AWS Toolkit for Visual Studio Code	4 marzo 2021
Nuova caratteristica	AWS Step Functions ora si integra con. AWS Glue DataBrew Per ulteriori informazioni, consultare: • Gestisci i AWS Glue DataBrew lavori con Step Functions • Che cos'è AWS Glue DataBrew? nella guida per gli DataBrew sviluppatori.	6 gennaio 2021

Modifica	Descrizione	Data della modifica
Nuova caratteristica	AWS Step Functions I flussi di lavoro Synchronous Express sono ora disponibili e offrono un modo semplice per orchestrare i microservizi. Per ulteriori informazioni, consultare: • Flussi di lavoro Express sincroni e asincroni • Un progetto di esempio che mostra come Richiama flussi di lavoro Synchronous Express • La documentazione StartSyncExecutiondell'API.	24 novembre 2020
Nuova caratteristica	AWS Step Functions ora si integra con Amazon API Gateway. Per ulteriori informazioni, consultare: • Chiama API Gateway con Step Functions • Politiche IAM per Step Functions e Politiche IAM per Amazon API Gateway • Un progetto di esempio che mostra come Effettuare una chiamata a API Gateway	17 novembre 2020
Nuova caratteristica	AWS Step Functions ora si integra con Amazon Elastic Kubernetes Service. Per ulteriori informazioni, consultare: • Chiama Amazon EKS con Step Functions • Politiche IAM per Step Functions e Politiche IAM per Amazon EKS • Un progetto di esempio che mostra come Gestisci un cluster Amazon EKS	16 Novembre 2020

Modifica	Descrizione	Data della modifica
Nuova caratteristica	AWS Step Functions ora si integra con Amazon Athena. Per ulteriori informazioni, consultare: • Chiama Athena con Step Functions • Politiche IAM per Step Functions e Politiche IAM per Amazon Athena • Un progetto di esempio che mostra come Avvia una query su Athena	22 ottobre 2020
Nuova caratteristica	AWS Step Functions ora supporta il tracciamento end-to-end dei flussi di lavoro AWS X-Ray, offrendovi una visibilità completa sulle esecuzioni delle macchine a stati e semplificando l'analisi e il debug delle applicazioni distribuite. Per ulteriori informazioni, consultare: • AWS X-Ray e Step Functions • Politiche IAM per Step Functions e Politiche IAM per AWS X-Ray • AWS Step Functions Documentazione di riferimento delle API • TracingConfiguration	14 settembre 2020

Modifica	Descrizione	Data della modifica

Aggiornamento	AWS Step Functions ora supporta payload di dimensioni fino a 256 KB di dati come stringa con codifica UTF-8. Ciò consente di elaborare payload più grandi nei flussi di lavoro Standard ed Express.
	
	Non è necessario modificare le macchine a stati esistenti per utilizzare i payload più grandi. Tuttavia, dovrai eseguire l'aggiornamento alle versioni più recenti di Step Functions SDK e Local Runner per utilizzare le API aggiornate. Per ulteriori informazioni, consultare:
	
	• Quote
	• the section called “Usa gli ARN Amazon S3 invece di trasferire payload di grandi dimensioni”
	• `States.DataLimitExceeded`
	• the section called “CloudWatchRegistra i payload”
	• the section called “EventBridge carichi utili”
	• AWS Step Functions Documentazione di riferimento delle API
	
	  • CloudWatchEventsExecutionDataDetails
	  • HistoryEventExecutionDataDetails
	  • GetExecutionHistory
	  • ActivityScheduledEventDetails
	  • ActivitySucceededEventDetails
	  • CloudWatchEventsExecutionDataDetails
	  • ExecutionSucceededEventDetails
	  • LambdaFunctionScheduledEventDetails
	  • ExecutionSucceededEventDetails
	  • StateEnteredEventDetails
	  • StateExitedEventDetails
		3 settembre 2020

Modifica	Descrizione	Data della modifica
	• TaskSubmittedEventDetails • TaskSucceededEventDetails	

Modifica	Descrizione	Data della modifica

Aggiornamento	L'Amazon States Language è stato aggiornato come segue:
	• Regole Choiceha aggiunto
	  • Un operatore di confronto nullo,`IsNull`. `IsNull`verifica il valore nullo JSON e può essere utilizzato per rilevare se l'output di uno stato precedent e è nullo o meno.
	  • Sono stati aggiunti altri quattro nuovi operatori, IsBoolean, IsNumeric e. IsString IsTimestamp
	  • Un test per l'esistenza o l'inesistenza di un campo utilizzando l'`IsPresent` operatore. `IsPresent` può essere usato per prevenire `States.Runtime` errori quando si tenta di accedere a una chiave inesistente.
	  • Corrispondenza di caratteri jolly per supportare il confronto tra stringhe e modelli con uno o più caratteri jolly.
	  • Confronto tra due variabili per gli operatori di confronto supportati.
	• I valori di timeout e heartbeat in uno `Task` stato possono ora essere forniti dinamicamente dall'input dello stato anziché un valore fisso utilizzando i `TimeoutSe condsPath` campi and. `HeartbeatSecondsPath` Per ulteriori informazioni, consulta Attività lo stato.
	• Il nuovo ResultSelector campo fornisce un modo per manipolare il risultato di uno stato prima che `ResultPat h` venga applicato. Il `ResultSelector` campo è un campo opzionale negli Attività stati Eseguire la mappaturaParallel, e.
	• Funzioni intrinsechesono stati aggiunti per consentir e operazioni di base senza `Task` stati. Le funzioni
		13 agosto 2020

Modifica	Descrizione	Data della modifica
	intrinseche possono essere utilizzate all'interno dei campi `Parameters` and`ResultSelector` .	
Aggiornamento	AWS Step Functions ora supporta la chiamata SageMaker `CreateProcessingJob` API Amazon. Per ulteriori informazioni, consultare: • Gestisci SageMaker con Step Functions • Preelabora i dati e addestra un modello di machine learning, un esempio di progetto che dimostra. `CreateProcessingJob`	4 agosto 2020
Nuova caratteristica	AWS Step Functions è ora supportato da AWS Serverles s Application Model, che semplifica l'integrazione dell'orch estrazione del flusso di lavoro nelle applicazioni serverless. Per ulteriori informazioni, consultare: • AWS Step Functions e AWS SAM • AWS::Serverless::StateMachine • AWS SAM Modelli di policy	27 maggio 2020
Nuova caratteristica	AWS Step Functions ha introdotto una nuova chiamata sincrona per annidare le esecuzioni di Step Functions. La nuova chiamata, `arn:aws:states:::states:sta rtExecution.sync:2` , restituisce un oggetto JSON. L'invocazione originale, `arn:aws:states:::s tates:startExecution.sync` , continua a essere supportata e restituisce una stringa di escape JSON. Per ulteriori informazioni, consultare: • Gestisci AWS Step Functions le esecuzioni come servizio integrato	19 maggio 2020

Modifica	Descrizione	Data della modifica
Nuova caratteristica	AWS Step Functions AWS CodeBuild ora si integra con. Per ulteriori informazioni, consultare: • Utilizzo AWS Step Functions con altri servizi • Chiama AWS CodeBuild con Step Functions • Integrazioni ottimizzate per Step Functions	5 maggio 2020
Nuova caratteristica	Step Functions è ora supportato in AWS Toolkit for Visual Studio Code, semplificando la creazione e la visualizz azione di flussi di lavoro basati su macchine a stati senza uscire dall'editor di codice.	31 marzo 2020
Aggiornamento	Ora puoi configurare la registrazione su Amazon CloudWatch Logs for Standard Workflow. Per ulteriori informazioni, consultare: • Registrazione utilizzandoCloudWatchRegistri	25 febbraio 2020
Nuova caratteristica	AWS Step Functions ora è possibile accedervi senza richiedere un indirizzo IP pubblico, direttamente da Amazon Virtual Private Cloud (VPC). Per ulteriori informazioni, consultare: • Endpoint Amazon VPC per Step Functions	23 dicembre 2019

Modifica	Descrizione	Data della modifica

Nuova caratteristica	I flussi di lavoro Express sono un nuovo tipo di flusso di lavoro, ideale per carichi di lavoro di elaborazione eventi a volume elevato, come l'acquisizione di dati IoT, l'elaborazione e la trasformazione dei dati in streaming e i back-end delle applicazioni mobili.
	
	Per ulteriori informazioni, esamina i seguenti argomenti nuovi e aggiornati.
	
	• Flussi di lavoro Standard e Express
	  • Garanzie di esecuzione
	• Utilizzo AWS Step Functions con altri servizi
	  • Integrazioni ottimizzate per Step Functions
	• Elabora messaggi ad alto volume da Amazon SQS (Express Workflows)
	• Esempio di checkpoint selettivo (flussi di lavoro Express)
	• Quote
	  • Quote
	• Registrazione utilizzandoCloudWatchRegistri
	• AWS Step Functions Documentazione di riferimento delle API
	  • CreateStateMachine
	  • UpdateStateMachine
	  • DescribeStateMachine
	  • DescribeStateMachineForExecution
	  • StopExecution
	  • DescribeExecution
	  • GetExecutionHistory
	  • ListExecutions
	  • ListStateMachines
		3 dicembre 2019

Modifica	Descrizione	Data della modifica
	• StartExecution • CloudWatchLogsLogGroup • LogDestination • LoggingConfiguration	
Nuova caratteristica	AWS Step Functions ora si integra con Amazon EMR. Per ulteriori informazioni, consultare: • Utilizzo AWS Step Functions con altri servizi • Chiama Amazon EMR con Step Functions • Integrazioni ottimizzate per Step Functions	19 novembre 2019
Aggiornamento	AWS Step Functions ha rilasciato AWS Step Functions Data Science SDK. Per ulteriori informazioni, consulta gli argomenti seguenti. • Progetto ONNX su GitHub • Documentazione sugli SDK • I seguenti taccuini di esempio, disponibili nella SageMakerconsole e nel relativo progetto. GitHub • `hello_world_workflow.ipynb` • `machine_learning_workflow_abalone.ipynb` • `training_pipeline_pytorch_mnist.ipynb`	7 novembre 2019

Modifica	Descrizione	Data della modifica
Aggiornamento	Step Functions ora supporta più azioni API per Amazon SageMaker e include due nuovi progetti di esempio per dimostrarne la funzionalità. Per ulteriori informazioni, consulta gli argomenti seguenti. • Gestisci SageMaker con Step Functions • Utilizzo AWS Step Functions con altri servizi • Training su un modello di Machine Learning • Ottimizzazione di un modello di Machine Learning.	3 ottobre 2019
Nuova caratteristica	Step Functions supporta l'avvio di nuove esecuzioni di flussi di lavoro `StartExecution` chiamando un'API di servizio integrata. Vedere: • Avvio delle esecuzioni di flusso di lavoro da uno stato di task • Gestisci AWS Step Functions le esecuzioni come servizio integrato • Utilizzo AWS Step Functions con altri servizi • Policy IAM per l'avvio delle esecuzioni dei flussi di lavoro di Step Functions	12 agosto 2019

Modifica	Descrizione	Data della modifica
Nuova caratteristica	Step Functions include la possibilità di passare un token di attività ai servizi integrati e di sospendere l'esecuzione fino a quando tale token di attività non viene restituito con `SendTaskSuccess` o `SendTaskFailure` . Vedere: • Modelli di integrazione dei servizi • Attendere un callback con il token dell'attività • Esempio di pattern di callback (Amazon SQS, Amazon SNS, Lambda) • Integrazioni ottimizzate per Step Functions • Distribuire un progetto di approvazione umana di esempio • Parametri di integrazione del servizio Step Functions ora offre un modo per accedere alle informazioni dinamiche sull'esecuzione corrente direttamente nel `"Parameters"` campo della definizione dello stato. Vedere: • Oggetto Context • Passa nodi di oggetti contestuali come parametri	23 maggio 2019
Nuova caratteristica	Step Functions supporta CloudWatch gli eventi per modificare lo stato di esecuzione, vedi: • EventBridge (CloudWatch Eventi) per le modifiche allo stato di esecuzione di Step Functions • Guida per l'utente di Amazon CloudWatch Events	8 maggio 2019

Modifica	Descrizione	Data della modifica
Nuova caratteristica	Step Functions supporta le autorizzazioni IAM tramite tag. Per ulteriori informazioni, consultare: • Inserimento di tag in Step Functions • Policy basate su tag	5 marzo 2019
Nuova caratteristica	Step Functions Local è ora disponibile. Puoi eseguire Step Functions sul tuo computer locale per test e sviluppo. Step Functions Local è disponibile per il download come applicazione Java o come immagine Docker. Per informazioni, consulta Test di macchine a stato locale.	4 febbraio 2019
Nuova caratteristica	AWS Step Functions è ora disponibile nelle regioni di Pechino e Ningxia. Per informazioni, consulta Regioni supportate.	15 gennaio 2018
Nuova caratteristica	Step Functions supporta l'etichettatura delle risorse per aiutarti a tenere traccia dell'allocazione dei costi. È possibile applicare tag alle macchine a stati nella pagina Details (Dettagli) o tramite operazioni API. Per informazioni, consulta Inserimento di tag in Step Functions.	7 gennaio 2019
Nuova caratteristica	AWS Step Functions è ora disponibile nelle regioni Europa (Parigi) e Sud America (San Paolo). Per informazioni, consulta Regioni supportate.	13 dicembre 2018
Nuova caratteristica	AWS Step Functions è ora disponibile nella regione Europa (Stoccolma). Per un elenco delle regioni supportate, consultare Regioni supportate.	12 dicembre 2018

Modifica	Descrizione	Data della modifica
Nuova caratteristica	Step Functions ora si integra con alcuni AWS servizi. Ora puoi chiamare e passare direttamente i parametri all'API di questi servizi integrati da uno stato di attività in Amazon States Language. Per ulteriori informazioni, consultare: • Utilizzo AWS Step Functions con altri servizi • Passa i parametri a un'API di servizio • Integrazioni ottimizzate per Step Functions	29 novembre 2018
Aggiornamento	Migliorata la descrizione di `TimeoutSeconds` e `HeartbeatSeconds` nella documentazione relativa agli stati task. Per informazioni, consulta Attività.	24 ottobre 2018
Aggiornamento	Migliorata la descrizione del limite Dimensione massima della cronologia di esecuzione e inserito un link all'argom ento di best practice correlato. • Quote relative alle esecuzioni di macchine a stati • Evita di raggiungere la quota di cronologia	17 ottobre 2018
Aggiornamento	È stato aggiunto un nuovo tutorial alla AWS Step Functions documentazione: VediAvvio dell'esecuzione di una macchina a stati in risposta agli eventi di Amazon S3.	25 settembre 2018
Aggiornamento	Rimossa la voce relativa al numero massimo di esecuzioni visualizzato nella console Step Functions dalla documenta zione sui limiti. Per informazioni, consulta Quote.	13 settembre 2018
Aggiornamento	È stato aggiunto un argomento sulle migliori pratiche alla AWS Step Functions documentazione sul miglioramento della latenza durante il polling delle attività. Per informazi oni, consulta Evita la latenza durante i sondaggi relativi alle attività.	30 agosto 2018

Modifica	Descrizione	Data della modifica
Aggiornamento	È stato migliorato l' AWS Step Functions argomento sulle attività e sugli addetti alle attività. Per informazioni, consulta Attività.	29 agosto 2018
Aggiornamento	È stato migliorato l' AWS Step Functions argomento sull' CloudTrail integrazione. Per informazioni, consulta Funzioni della fase di registrazione tramite AWS CloudTrail.	7 agosto 2018
Aggiornamento	Aggiunti esempi JSON al AWS CloudFormation tutorial. Per informazioni, consulta Creazione di una macchina a stati Lambda per Step Functions utilizzando AWS CloudFormation.	23 giugno 2018
Aggiornamento	È stato aggiunto un nuovo argomento sulla gestione degli errori del servizio Lambda. Per informazioni, consulta Gestire le eccezioni del servizio Lambda.	20 giugno 2018
Nuova caratteristica	AWS Step Functions è ora disponibile nella regione Asia Pacifico (Mumbai). Per un elenco delle regioni supportate, consultare Regioni supportate.	28 giugno 2018
Nuova caratteristica	AWS Step Functions è ora disponibile la regione AWS GovCloud (Stati Uniti occidentali). Per un elenco delle regioni supportate, consultare Regioni supportate. Per informazioni sull'utilizzo di Step Functions nella regione AWS GovCloud (Stati Uniti occidentali), vedere AWS GovCloud (US).	28 giugno 2018
Aggiornamento	Migliorata la documentazione sulla gestione degli errori per gli stati `Parallel`. Per informazioni, consulta Gestione errori.	20 giugno 2018

Modifica	Descrizione	Data della modifica
Aggiornamento	Documentazione migliorata sull'elaborazione di input e output in Step Functions. Viene illustrato come utilizzar e `InputPath` , `ResultPath` e `OutputPath` per controllare il flusso di JSON nei flussi di lavoro, negli stati e nei task. Vedere: • Elaborazione di input e output in Step Functions • ResultPath	7 giugno 2018
Aggiornamento	Migliorati gli esempi di codice per gli stati Parallel. Per informazioni, consulta Parallel.	4 giugno 2018
Nuova caratteristica	Ora puoi monitorare le metriche delle API e dei servizi in CloudWatch. Per informazioni, consulta Monitoraggio di Step Functions tramite CloudWatch.	25 maggio 2018
Aggiornamento	I limiti di throttling di `StartExecution` , `StopExecution` e `StateTransition` sono stati aumentati nelle regioni seguenti: • Stati Uniti orientali (Virginia settentrionale) • US West (Oregon) • Europa (Irlanda) Per ulteriori informazioni, consulta Quote.	16 maggio 2018
Nuova caratteristica	AWS Step Functions è ora disponibile nelle regioni Stati Uniti occidentali (California settentrionale) e Asia Pacifico (Seoul). Per un elenco delle regioni supportate, consultare Regioni supportate.	5 maggio 2018
Aggiornamento	Aggiornate le procedure e le immagini in base alle modifiche apportate all'interfaccia.	25 aprile 2018

Modifica	Descrizione	Data della modifica
Aggiornamento	Aggiunto un nuovo tutorial che illustra come avviare una nuova esecuzione per continuare il lavoro. Per informazioni, consulta Esecuzioni continue di flussi di lavoro a lungo termine come nuova esecuzione. Il tutorial descrive un modello di progettazione che consente di evitare alcune limitazioni del servizio. Per informazioni, consulta Evita di raggiungere la quota di cronologia.	19 aprile 2018
Aggiornamento	Migliorata la documentazione introduttiva sugli stati con l'aggiunta di informazioni concettuali sulle macchine a stati. Per informazioni, consulta States.	9 marzo 2018
Aggiornamento	Oltre a HTML, PDF e Kindle, la Guida per gli AWS Step Functions sviluppatori è disponibile su. GitHub Per lasciare un feedback, scegli l' GitHubicona nell'angolo in alto a destra.	2 marzo 2018
Aggiornamento	È stato aggiunto un argomento che descrive altre risorse relative a Step Functions. Per informazioni, consulta Informazioni correlate.	20 febbraio 2018

Modifica	Descrizione	Data della modifica

Nuova caratteristica	• Quando crei una nuova macchina a stati, devi riconosce re che AWS Step Functions verrà creato un ruolo IAM che consente l'accesso alle tue funzioni Lambda.
	• Aggiornati i tutorial seguenti per riflettere le modifiche minori nel flusso di lavoro per la creazione di una macchina a stati:
	  • Creazione di una macchina a stati Step Functions che utilizza Lambda
	  • Creazione di una macchina a stati Activity utilizzando Step Functions
	  • Gestione delle condizioni di errore utilizzando una macchina a stati Step Functions
	  • Iterare un ciclo con Lambda
		19 febbraio 2018

Aggiornamento	Aggiunto un argomento che illustra un worker di attività di esempio scritto in Ruby. Questa implementazione può essere utilizzata per creare direttamente un worker di attività Ruby oppure come modello di progettazione per creare un worker di attività in un altro linguaggio. Per informazioni, consulta Esempio di lavoratore di attività in Ruby.	6 febbraio 2018
Aggiornamento	È stato aggiunto un nuovo tutorial che descrive un modello di progettazione che utilizza una funzione Lambda per iterare un conteggio. Per informazioni, consulta Creazione di una macchina a stati Step Functions che utilizza Lambda.	31 gennaio 2018

Modifica	Descrizione	Data della modifica
Aggiornamento	Contenuto aggiornato sulle autorizzazioni IAM da includere e sulle API. `DescribeStateMachineForExecution` `UpdateStateMachine` Per informazioni, consulta Creazione di autorizzazioni IAM granulari per utenti non amministratori.	26 gennaio 2018
Aggiornamento	Sono state aggiunte nuove regioni disponibili: Canada (Centrale), Asia Pacifico (Singapore). Per informazioni, consulta Regioni supportate.	25 gennaio 2018
Aggiornamento	Tutorial e procedure aggiornati per dimostrare che IAM consente di selezionare Step Functions come ruolo.	24 gennaio 2018
Aggiornamento	Aggiunto un nuovo argomento di best practice che suggerisce di non passare payload di grandi dimensioni tra gli stati. Per informazioni, consulta Usa gli ARN Amazon S3 invece di trasferire payload di grandi dimensioni.	23 gennaio 2018
Aggiornamento	Corrette le procedure in base all'aggiornamento dell'interfaccia per la creazione di una macchina a stati: • Creazione di una macchina a stati Step Functions che utilizza Lambda • Creazione di una macchina a stati Activity utilizzando Step Functions • Gestione delle condizioni di errore utilizzando una macchina a stati Step Functions	17 gennaio 2018

Modifica	Descrizione	Data della modifica
Nuova caratteristica	È possibile utilizzare progetti di esempio per effettuare rapidamente il provisioning delle macchine a stati e di tutte le risorse AWS correlate. Consultare Progetti di esempio per Step Functions. I progetti di esempio disponibili includono: • Sondaggio per Job Status (AWS Batch Lambda,) • Timer delle attività (Lambda, Amazon SNS) **Note** Questi progetti di esempio e la documentazione correlata sostituiscono i tutorial in cui veniva illustrat a l'implementazione delle stesse funzionalità.	11 gennaio 2018
Aggiornamento	Aggiunta una sezione di best practice contenente informazi oni su come evitare il blocco delle esecuzioni. Per informazi oni, consulta Le migliori pratiche per Step Functions.	5 gennaio 2018
Aggiornamento	È stata aggiunta una nota su come i nuovi tentativi possono influire sui prezzi: **Note** I nuovi tentativi vengono trattati come transizioni di stato. Per informazioni su come le transizioni di stato influiscono sulla fatturazione, consulta Step Functions Pricing.	8 dicembre 2017

Modifica	Descrizione	Data della modifica
Aggiornamento	Aggiunte informazioni relative ai nomi delle risorse. **Note** Step Functions consente di creare nomi per macchine a stati, esecuzioni e attività ed etichette che contengono caratteri non ASCII. Questi nomi non ASCII non funzionano con Amazon. CloudWatch Per assicurarti di poter tenere traccia delle CloudWatch metriche, scegli un nome che utilizzi solo caratteri ASCII.	6 dicembre 2017
Aggiornamento	Informazioni di panoramica sulla sicurezza migliorate e aggiunto un argomento sulle autorizzazioni IAM granulari. Consulta Sicurezza in AWS Step Functions e Creazione di autorizzazioni IAM granulari per utenti non amministratori.	27 Novembre 2017
Nuova caratteristica	È possibile aggiornare una macchina a stati esistente. Vedi Aggiorna la tua macchina a stati.	15 novembre 2017

Modifica	Descrizione	Data della modifica
Aggiornamento	Aggiunta una nota per chiarire gli errori `Lambda.Unknown` con link alla documentazione di Lambda nelle sezioni seguenti: • Nomi degli errori • Fase 3: Creare una macchina a stati con un campo Catch **Note** Gli errori non gestiti in Lambda vengono riportati come `Lambda.Unknown` nell'output degli errori. Questi includono out-of-memory errori e timeout delle funzioni. Puoi abbinare o `States.TaskFailed` gestire questi errori. `Lambda.Unknown` `States.ALL` Quando Lambda raggiunge il numero massimo di chiamate, l'errore è. `Lambda.TooManyRequestsException` Per ulteriori informazioni sugli errori della funzione Lambda, consulta Gestione degli errori e tentativi automatici nella Developer Guide.AWS Lambda	17 ottobre 2017
Aggiornamento	Istruzioni IAM corrette e chiarite e aggiornate le schermate di tutti i tutorial.	11 ottobre 2017

Modifica	Descrizione	Data della modifica

Aggiornamento	• Sono state aggiunte nuove schermate per i risultati di esecuzione delle macchine a stati per riflettere le modifiche nella console Step Functions. Ha riscritto le istruzioni Lambda nei seguenti tutorial per riflettere le modifiche nella console Lambda:
	  • Creazione di una macchina a stati Step Functions che utilizza Lambda
	  • Creazione di un poller per lo stato del processo
	  • Creazione di un timer di task
	  • Gestione delle condizioni di errore utilizzando una macchina a stati Step Functions
	• Corrette e chiarite le informazioni sulla creazione di macchine a stati nelle sezioni seguenti:
	  • Creazione di una macchina a stati Activity utilizzando Step Functions
		6 ottobre 2017

Aggiornamento	Ha riscritto le istruzioni IAM nelle seguenti sezioni per riflettere le modifiche nella console IAM: • Creazione di un ruolo IAM per la tua macchina a stati • Creazione di una macchina a stati Step Functions che utilizza Lambda • Creazione di un poller per lo stato del processo • Creazione di un timer di task • Gestione delle condizioni di errore utilizzando una macchina a stati Step Functions • Creazione di un'API Step Functions utilizzando API Gateway	5 ottobre 2017
Aggiornamento	Riscritta la sezione Dati delle macchine a stati.	28 settembre 2017

Modifica	Descrizione	Data della modifica
Nuova caratteristica	I limiti relativi alla limitazione delle azioni delle API sono aumentati per tutte le regioni in cui è disponibile Step Functions.	18 settembre 2017
Aggiornamento	• Corrette e chiarite le informazioni sull'avvio di nuove esecuzioni in tutti i tutorial. • Corrette e chiarite le informazioni nella sezione Quote relative agli account.	14 settembre 2017
Aggiornamento	Ha riscritto i seguenti tutorial per riflettere le modifiche nella console Lambda: • Creazione di una macchina a stati Step Functions che utilizza Lambda • Gestione delle condizioni di errore utilizzando una macchina a stati Step Functions • Creazione di un poller per lo stato del processo	28 agosto 2017
Nuova caratteristica	Step Functions è disponibile in Europa (Londra).	23 agosto 2017
Nuova caratteristica	I flussi di lavoro visivi delle macchine a stati consentono di ingrandire, rimpicciolire e centrare il grafico.	21 agosto 2017

Modifica	Descrizione	Data della modifica
Nuova caratteristica	**Important** Un'esecuzione non può utilizzare il nome di un'altra esecuzione per 90 giorni. Quando si effettuano più `StartExecution` chiamate con lo stesso nome, la nuova esecuzione non viene eseguita. Per ulteriori informazioni, consulta il parametro di `name`richiesta dell'azione `StartExecution` API nell'AWS Step Functions API Reference.	18 agosto 2017
Aggiornamento	Aggiunte informazioni su un modo alternativo per passare l'ARN della macchina a stati nel tutorial Creazione di un'API Step Functions utilizzando API Gateway.	17 agosto 2017
Aggiornamento	Aggiunto il nuovo tutorial sulla creazione di un poller per lo stato del processo.	10 agosto 2017
Nuova caratteristica	• Step Functions emette la `ExecutionThrottled` CloudWatch metrica. Per ulteriori informazioni, consulta Monitoraggio di Step Functions tramite CloudWatch. • Aggiunta la sezione Quote relative alla limitazione statale.	3 agosto 2017
Aggiornamento	Istruzioni nella sezione Fase 1: creazione di un ruolo IAM per API Gateway aggiornate.	18 luglio 2017
Aggiornamento	Corrette e chiarite le informazioni nella sezione Choice.	23 giugno 2017

Modifica	Descrizione	Data della modifica
Aggiornamento	Sono state aggiunte informazioni sull'utilizzo delle risorse di altri AWS account ai seguenti tutorial: • Creazione di una macchina a stati Step Functions che utilizza Lambda • Creazione di una macchina a stati Lambda per Step Functions utilizzando AWS CloudFormation • Creazione di una macchina a stati Activity utilizzando Step Functions • Gestione delle condizioni di errore utilizzando una macchina a stati Step Functions	22 giugno 2017
Aggiornamento	Corrette e chiarite le informazioni nelle sezioni seguenti: • Gestione delle condizioni di errore utilizzando una macchina a stati Step Functions • States • Gestione degli errori in Step Functions	21 giugno 2017
Aggiornamento	Ha riscritto tutti i tutorial in modo che corrispondano all'aggiornamento della console Step Functions.	12 giugno 2017
Nuova caratteristica	Step Functions è disponibile in Asia Pacifico (Sydney).	8 giugno 2017
Aggiornamento	Ristrutturata la sezione Amazon States Language.	7 giugno 2017
Aggiornamento	Corrette e chiarite le informazioni nella sezione Creazione di una macchina a stati Activity utilizzando Step Functions.	6 giugno 2017
Aggiornamento	Corretti gli esempi di codice nella sezione Esempi di macchine a stati che utilizzano Retry e Catch.	5 giugno 2017

Modifica	Descrizione	Data della modifica
Aggiornamento	Ha ristrutturato questa guida utilizzando standard di AWS documentazione.	31 maggio 2017
Aggiornamento	Corrette e chiarite le informazioni nella sezione Parallel.	25 maggio 2017
Aggiornamento	Unite le sezioni relative a percorsi e filtri nella sezione Elaborazione di input e output in Step Functions.	24 maggio 2017
Aggiornamento	Corrette e chiarite le informazioni nella sezione Monitoraggio di Step Functions tramite CloudWatch.	15 maggio 2017
Aggiornamento	Aggiornato il codice del worker `GreeterActivities.java` nel tutorial Creazione di una macchina a stati Activity utilizzando Step Functions.	9 maggio 2017
Aggiornamento	Aggiunto un video introduttivo alla sezione Che cos'è AWS Step Functions?.	19 aprile 2017
Aggiornamento	Corrette e chiarite le informazioni nei tutorial seguenti: • Creazione di una macchina a stati Step Functions che utilizza Lambda • Creazione di una macchina a stati Activity utilizzando Step Functions • Gestione delle condizioni di errore utilizzando una macchina a stati Step Functions	19 aprile 2017

Modifica	Descrizione	Data della modifica
Aggiornamento	Sono state aggiunte informazioni sui modelli Lambda Creazione di una macchina a stati Step Functions che utilizza Lambda e Gestione delle condizioni di errore utilizzando una macchina a stati Step Functions sui tutorial.	6 Aprile 2017
Aggiornamento	Modificato il limite della dimensione massima dei dati di input o risultati nel limite di dimensione massima dei dati di input o risultati per un task, uno stato o un'esecuzione (32.768 caratteri). Per ulteriori informazioni, consulta Quote relative all'esecuzione delle attività.	31 marzo 2017
Nuova caratteristica	• Step Functions supporta l'esecuzione di macchine a stati impostando Step Functions come obiettivi di Amazon CloudWatch Events.	21 marzo 2017
Nuova caratteristica	• Step Functions consente la gestione degli errori della funzione Lambda come metodo di gestione degli errori preferito. • Aggiornati il tutorial Gestione delle condizioni di errore utilizzando una macchina a stati Step Functions e la sezione Gestione degli errori in Step Functions.	16 marzo 2017
Nuova caratteristica	Step Functions è disponibile in Europa (Francoforte).	7 marzo 2017
Aggiornamento	Riorganizzati gli argomenti nel sommario e aggiornati i tutorial seguenti: • Creazione di una macchina a stati Step Functions che utilizza Lambda • Creazione di una macchina a stati Activity utilizzando Step Functions • Gestione delle condizioni di errore utilizzando una macchina a stati Step Functions	23 febbraio 2017

Modifica	Descrizione	Data della modifica
Nuova caratteristica	• La pagina State Machines della console Step Functions include i pulsanti Copia su nuovo ed Elimina. • Aggiornati gli screenshot in base alle modifiche della console.	23 febbraio 2017
Nuova caratteristica	• Step Functions supporta la creazione di API utilizzando API Gateway. • Aggiunto il tutorial Creazione di un'API Step Functions utilizzando API Gateway.	14 febbraio 2017
Nuova caratteristica	• Step Functions supporta l'integrazione con AWS CloudFormation. • Aggiunto il tutorial Creazione di una macchina a stati Lambda per Step Functions utilizzando AWS CloudFormation.	10 febbraio 2017
Aggiornamento	Chiarito il comportamento corrente dei campi `ResultPath` e `OutputPath` in relazione agli stati `Parallel`.	6 febbraio 2017
Aggiornamento	• Chiarite le limitazioni di denominazione delle macchine a stati nei tutorial. • Corretti alcuni esempi di codice.	5 gennaio 2017
Aggiornamento	Esempi di funzioni Lambda aggiornati per utilizzare il modello di programmazione più recente.	9 dicembre 2016
Rilascio iniziale	Versione iniziale di. AWS Step Functions	1° dicembre 2016

Glossario per AWS

Per la terminologia AWS più recente, consultare il glossario AWS nella documentazione di riferimento per Glossario AWS.

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.