



Automatizza la tua soluzione DR per database relazionali su AWS

AWS Linee guida prescrittive



AWS Linee guida prescrittive: Automatizza la tua soluzione DR per database relazionali su AWS

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

I marchi e l'immagine commerciale di Amazon non possono essere utilizzati in relazione a prodotti o servizi che non siano di Amazon, in una qualsiasi modalità che possa causare confusione tra i clienti o in una qualsiasi modalità che denigri o discrediti Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà dei rispettivi proprietari, che possono o meno essere affiliati, collegati o sponsorizzati da Amazon.

Table of Contents

Introduzione	1
Obiettivi	1
Panoramica di	3
Il processo di failover	4
Architettura DR Orchestrator FAILOVER	4
Il processo di failback	5
Architettura DR Orchestrator FAILBACK	5
FAILOVER DI DR Orchestrator	7
Dettagli dei parametri	8
FAILBACK DI DR Orchestrator	9
Stati Aurora e macchine a stati Step Functions	12
Stato stazionario	12
Stato dell'evento	12
Passaggio	13
Failover manuale non pianificato	18
Failback	22
Implementa la soluzione	31
Domande frequenti	33
Quali sono gli RPO e gli RTO che posso ottenere utilizzando questo approccio?	33
È obbligatorio utilizzare le variabili di AWS CloudFormation esportazione?	33
Posso utilizzare il flusso di lavoro FAILOVER di DR Orchestrator per eseguire il failover di più di un database? AWS	33
Come posso evitare l' InvalidParameterCombination errore quando eseguo la macchina a stati DR Orchestrator FAILBACK per Amazon RDS?	34
Passaggi successivi	35
Risorse	36
Cronologia dei documenti	37
.....	xxxviii

Automatizza la tua soluzione DR per database relazionali su AWS

Jitendra Kumar, Oliver Francis e Pavithra Balasubramanian, Amazon Web Services (AWS)

Maggio 2024 ([storia del documento](#))

Quando pianifichi una [strategia di disaster recovery \(DR\)](#) su Amazon Web Services (AWS), puoi implementare una soluzione DR per i database della tua organizzazione. Puoi automatizzare la soluzione DR configurando un'architettura basata sugli eventi per eseguire il failover delle istanze del database. I vantaggi di questo approccio includono quanto segue:

- Ridurre gli errori umani causati dal coordinamento manuale
- Fornire in alcuni casi il Recovery Time Objective (RTO) più breve possibile
- Facilitazione dei test ripetibili della soluzione DR

Senza automazione, un'interruzione del sistema primario Regione AWS richiederebbe che le risorse dell'organizzazione eseguissero manualmente i passaggi indicati in un run-book. Questi passaggi potrebbero includere più processi, come la promozione delle repliche di lettura di Amazon Relational Database Service (Amazon RDS) nel sistema Regione AWS secondario affinché fungano da nuova istanza di database principale. AWS fornisce [AWS SDK for Python \(Boto3\)](#) per API automatizzare queste azioni, che possono costituire gli elementi costitutivi di una soluzione DR per le esigenze dell'organizzazione.

Questa guida illustra un orchestratore DR centralizzato e basato su manifesti che può aiutarvi a realizzare una soluzione di DR per i database all'interno dello stesso AWS Account. La decisione di eseguire un failover richiederebbe comunque l'intervento umano e non rientra nell'ambito dell'automazione.

I destinatari di questa guida sono gli architetti delle applicazioni, gli architetti dell'infrastruttura, gli architetti di database e i consulenti di database.

Obiettivi

L'implementazione della soluzione DR Orchestrator Framework può aiutarti a raggiungere i seguenti risultati:

- Riduzione degli errori manuali durante un'interruzione: le prestazioni manuali dei processi di DR possono essere soggette a errori. L'automazione delle azioni durante le fasi di un'attività di DR riduce gli errori manuali e aiuta a garantire un failover o un failback più fluidi.
- Gestisci il DR su larga scala: se la tua organizzazione dispone di un'ampia flotta di database, l'automazione degli edifici per il DR ti aiuta a scalare la tua soluzione di DR. È possibile automatizzare la soluzione DR AWS del database utilizzando DR Orchestrator Framework. DR Orchestrator automatizza i passaggi per il failover o il failback dei database utilizzando un file di AWS parametri in formato JSON.

Panoramica di DR Orchestrator Framework

DR Orchestrator Framework offre una soluzione con un solo clic per orchestrare e automatizzare il ripristino di emergenza interregionale per i database. AWS Utilizza [AWS Step Functions](#) ed esegue i passaggi richiesti durante [AWS Lambda](#) il failover e il failback. Le macchine a stati Step Functions forniscono la base per il processo decisionale all'interno della progettazione dell'orchestratore. Le operazioni API per l'esecuzione di azioni di failover o failback sono codificate in funzioni Lambda richiamate dall'interno della macchina a stati. Le funzioni Lambda eseguono [AWS SDK per Python \(Boto3\)](#) API per interagire con i database. AWS

DR Orchestrator Framework contiene due macchine a stati principali che corrispondono alle fasi di failover e failback.

Per Amazon RDS, la fase di failover promuove una replica di lettura RDS interregionale in un'istanza DB autonoma. Per Amazon Aurora, quando la regione principale è inattiva durante un'interruzione rara e inaspettata, il relativo nodo di scrittura non è disponibile. La replica tra il nodo di scrittura e i cluster secondari si interrompe. È necessario scollegare il cluster secondario dal database globale e promuoverlo come cluster autonomo. Le applicazioni possono connettersi e inviare traffico di scrittura al cluster autonomo. È possibile utilizzare lo stesso processo per [passare](#) dal cluster DB primario del database globale alle regioni secondarie. Utilizzate questo approccio per scenari controllati come i seguenti:

- Manutenzione operativa
- Procedure operative pianificate
- Promozione di un cluster secondario Amazon ElastiCache (Redis OSS) come nuovo cluster primario

La fase di failback stabilisce la replica in tempo reale dei dati tra una regione primaria attiva e una nuova regione secondaria.

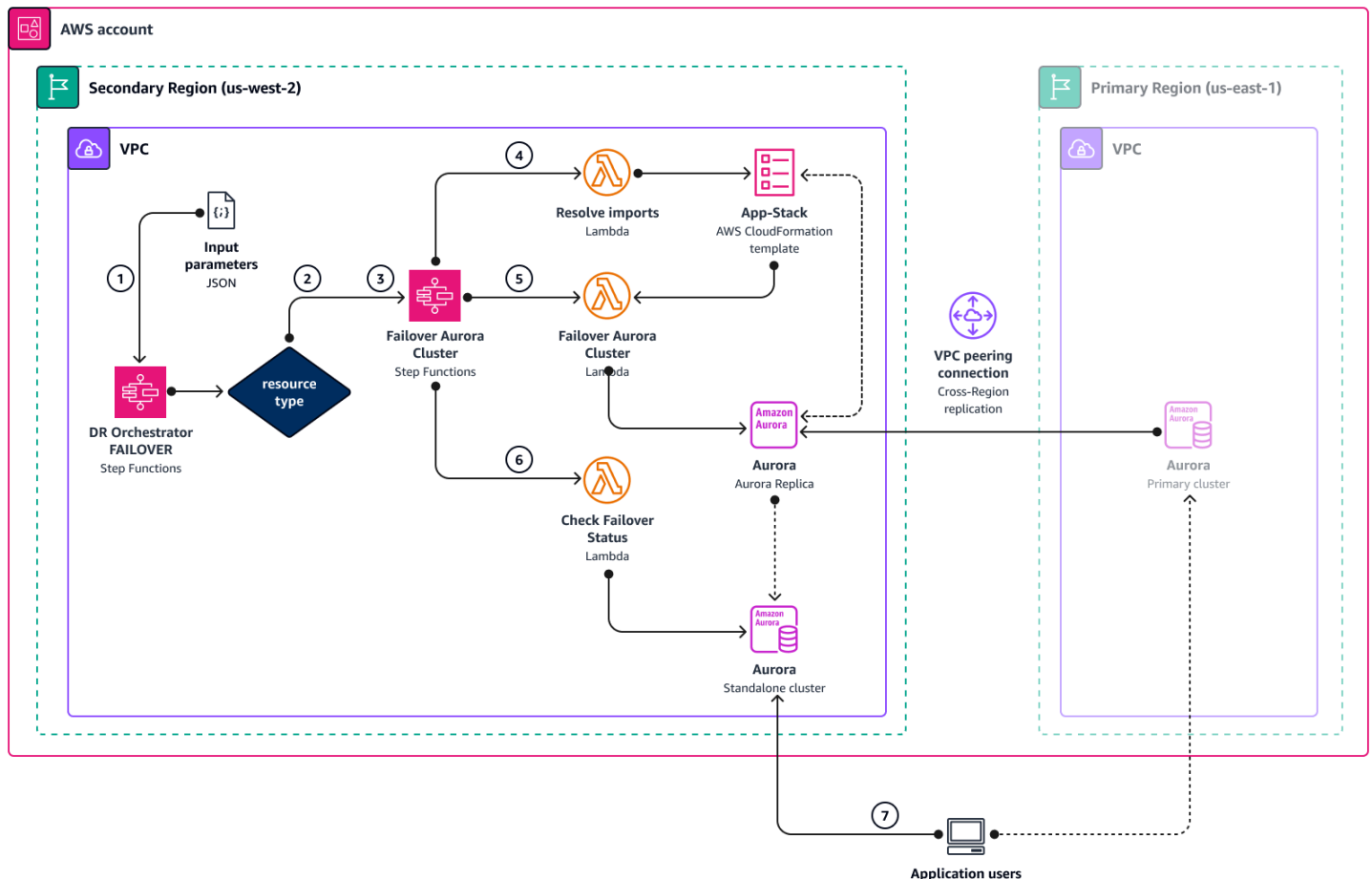
È fondamentale comprendere che DR Orchestrator si applica solo ai database. Tutte le applicazioni che fanno riferimento a questi database e si trovano nella stessa regione potrebbero richiedere una soluzione di failover tandem separata. Dopo il failover dei database nella regione secondaria, è necessario aggiornare le applicazioni per connettersi alle nuove istanze del database, che fungeranno da origine dati.

Il processo di failover

Per eseguire un failover, eseguite la macchina a DR `Orchestrator FAILOVER` stati. In questa fase, un database secondario è già presente nella regione secondaria, come replica di lettura (Amazon RDS) o come cluster secondario (Amazon Aurora). Quando si esegue la macchina a DR `Orchestrator FAILOVER` stati, il database secondario diventa il principale.

Architettura DR Orchestrator FAILOVER

Il diagramma seguente mostra i concetti del processo di failover per Amazon Aurora quando si utilizza DR Orchestrator. Amazon Aurora e Amazon ElastiCache utilizzano lo stesso flusso di lavoro ma con macchine a stati e funzioni Lambda diverse.



1. La macchina a DR `Orchestrator FAILOVER` stati legge i parametri JSON di input.
2. In base al `resourceType` parametro, la macchina a stati chiama altre macchine a stati: `Promote RDS Read ReplicaFailover Aurora Cluster`, o. `Failover ElastiCache`. Se nell'input viene passata più di una risorsa, queste macchine a stati funzionano in parallelo.

3. La macchina a Failover Aurora Cluster stati richiama le funzioni Lambda in ciascuno dei tre passaggi seguenti.
4. La funzione Resolve imports Lambda si risolve "! import <export-variable-name>" con i valori effettivi del modello. App-Stack AWS CloudFormation
5. La funzione **Failover Aurora Cluster** Lambda promuove la replica di lettura come istanza DB autonoma.
6. La funzione **Check Failover Status** Lambda controlla lo stato dell'istanza DB promossa. Dopo che lo stato è DISPONIBILE, la funzione Lambda invia un token di successo alla macchina a stati chiamante e completa.
7. È possibile reindirizzare le applicazioni al database autonomo nella regione DR (us-west-2), che ora è il database principale.

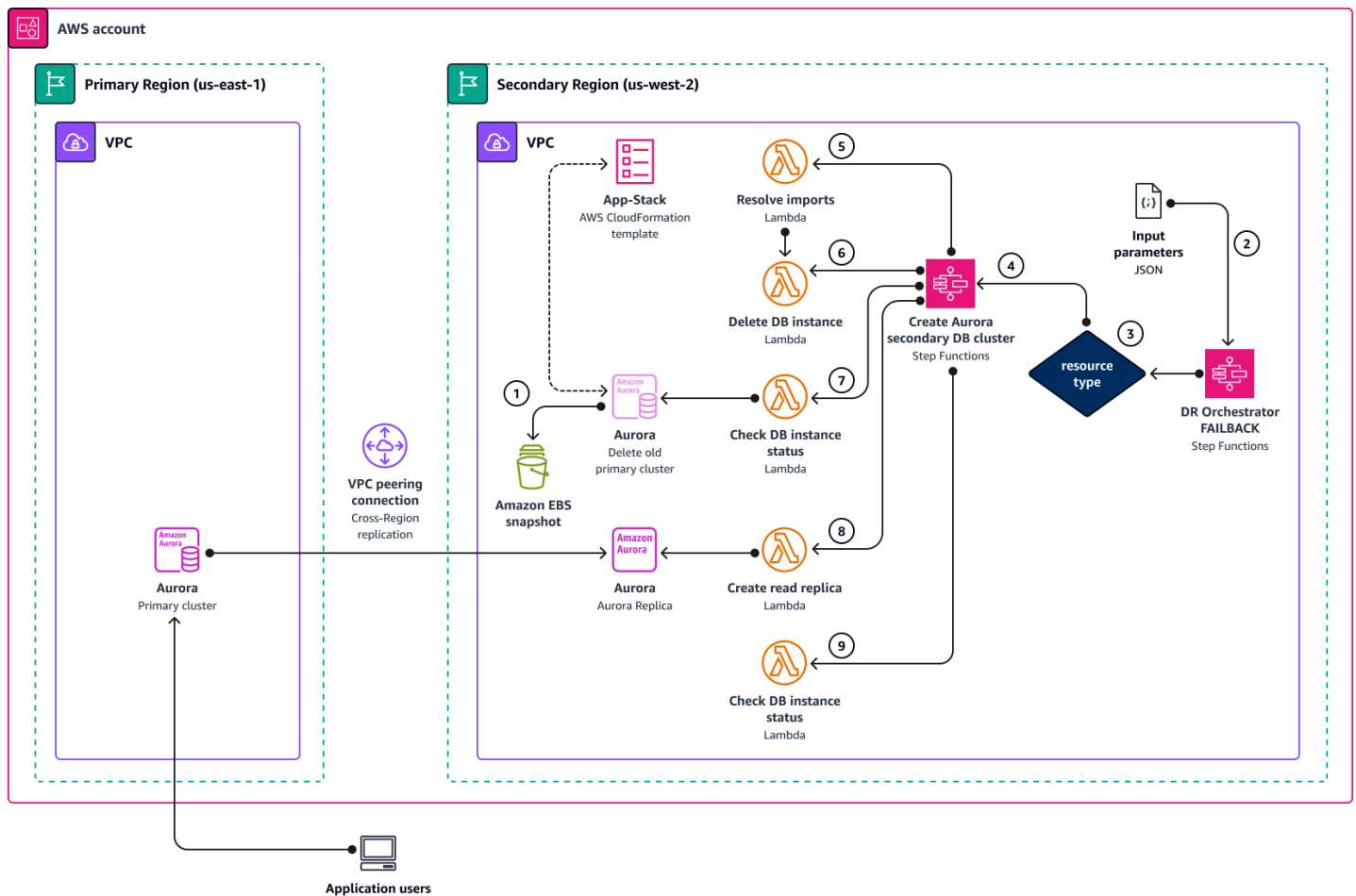
Il processo di failback

Dopo che la regione principale precedente (us-east-1) è nuovamente attiva, è possibile eseguire il failback, in modo che il database in entrata us-east-1 diventi nuovamente la principale. Per avviare il failback, esegui la macchina a DR Orchestrator FAILBACK stati. Come indica il nome, questa macchina a stati inizia a replicare le modifiche nella nuova regione principale (us-west-2) nella precedente regione primaria (us-east-1), che funge da regione secondaria corrente.

Una volta stabilita la replica tra le due regioni, è possibile avviare il failback. Per eseguire il failback e tornare alla regione primaria originale (us-east-1), esegui la macchina a DR Orchestrator FAILOVER stati nella regione secondaria corrente (us-east-1) per promuoverla nella regione primaria.

Architettura DR Orchestrator FAILBACK

Il diagramma seguente mostra i concetti del processo di failback per Amazon Aurora quando si utilizza DR Orchestrator.



1. Prima di iniziare il failback, scatta un'istantanea manuale del DB da utilizzare per eseguire l'analisi della causa principale (RCA).

Inoltre, disabilita il DeletionProtection per il cluster Aurora nella regione primaria precedente (us-east-1).

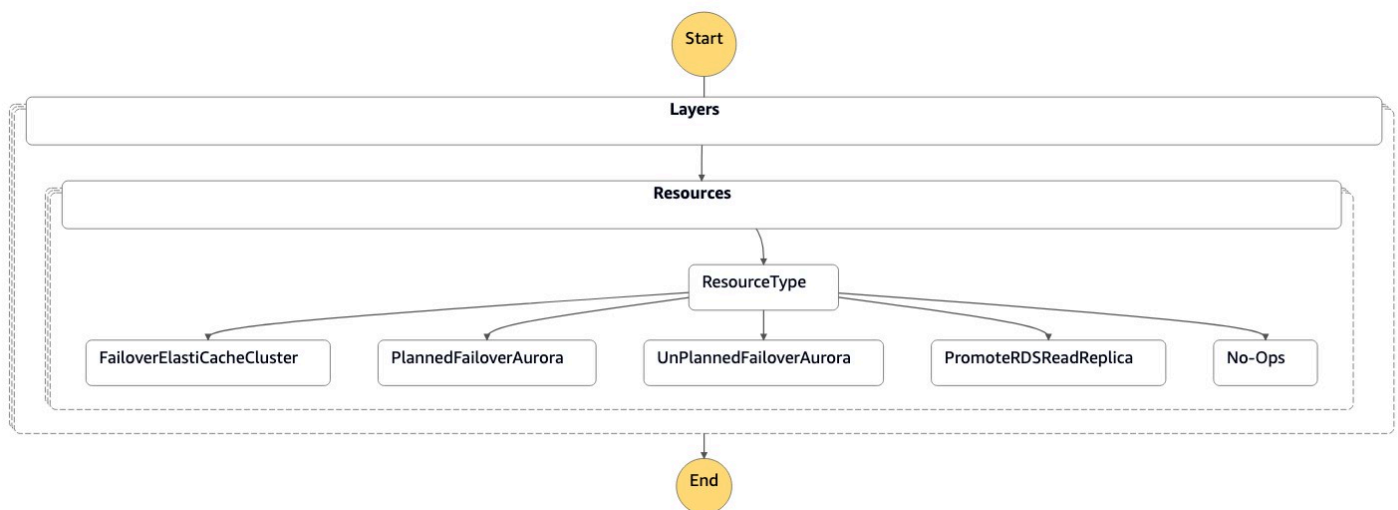
2. La macchina a DR Orchestrator FAILBACK stati legge i parametri JSON di input.
3. In base aresourceType, la macchina a DR Orchestrator FAILBACK stati chiama la macchina a Create Aurora Secondary DB Cluster stati.
4. La macchina a Create Aurora Secondary DB Cluster stati richiama le funzioni Lambda in ciascuno dei seguenti cinque passaggi.
5. La funzione Resolve import Lambda si risolve "! import <export-variable-name>" con i valori effettivi del modello. App-Stack CloudFormation
6. La funzione Delete DB Instance Lambda elimina la precedente istanza principale.

7. La funzione `Check DB instance status` Lambda controlla `Delete DB Instance status` fino all'eliminazione del DB.
8. La funzione `Create Read Replica` Lambda crea una replica di lettura nella regione secondaria dall'istanza DB che si trova nella nuova regione primaria.
9. La funzione `Check DB instance status` Lambda verifica lo stato dell'istanza DB di replica in lettura. Quando lo stato è `DISPONIBILE`, la funzione Lambda invia un token di successo alla macchina a stati chiamante, operazione completata.

FAILOVER DI DR Orchestrator

Utilizza la macchina a DR Orchestrator FAILOVER stati nell'evento DR quando la regione primaria (us-east-1) è inattiva o durante eventi pianificati come la manutenzione operativa.

La funzione può essere chiamata per eseguire il failover di uno o più database in parallelo.



La macchina a stati accetta parametri in formato JSON, come illustrato nel codice seguente:

```

{
  "StatePayload": [
    {
      "layer": 1,
      "resources": [
        {
          "resourceType": "PromoteRDSReadReplica",
          "resourceName": "Promote RDS MySQL Read Replica",
          "parameters": {

```

```

        "RDSInstanceIdentifier": "!Import rds-mysql-instance-identifier",
        "TargetClusterIdentifier": "!Import rds-mysql-instance-global-arn"
    },
    {
        "resourceType": "FailoverElastiCacheCluster",
        "resourceName": "Failover ElastiCache Cluster",
        "parameters": {
            "GlobalReplicationGroupId": "!Import demo-redis-cluster-global-replication-
group-id",
            "TargetRegion": "!Import demo-redis-cluster-target-region",
            "TargetReplicationGroupId": "!Import demo-redis-cluster-target-replication-
group-id"
        }
    }
]
}
]
}

```

Dettagli dei parametri

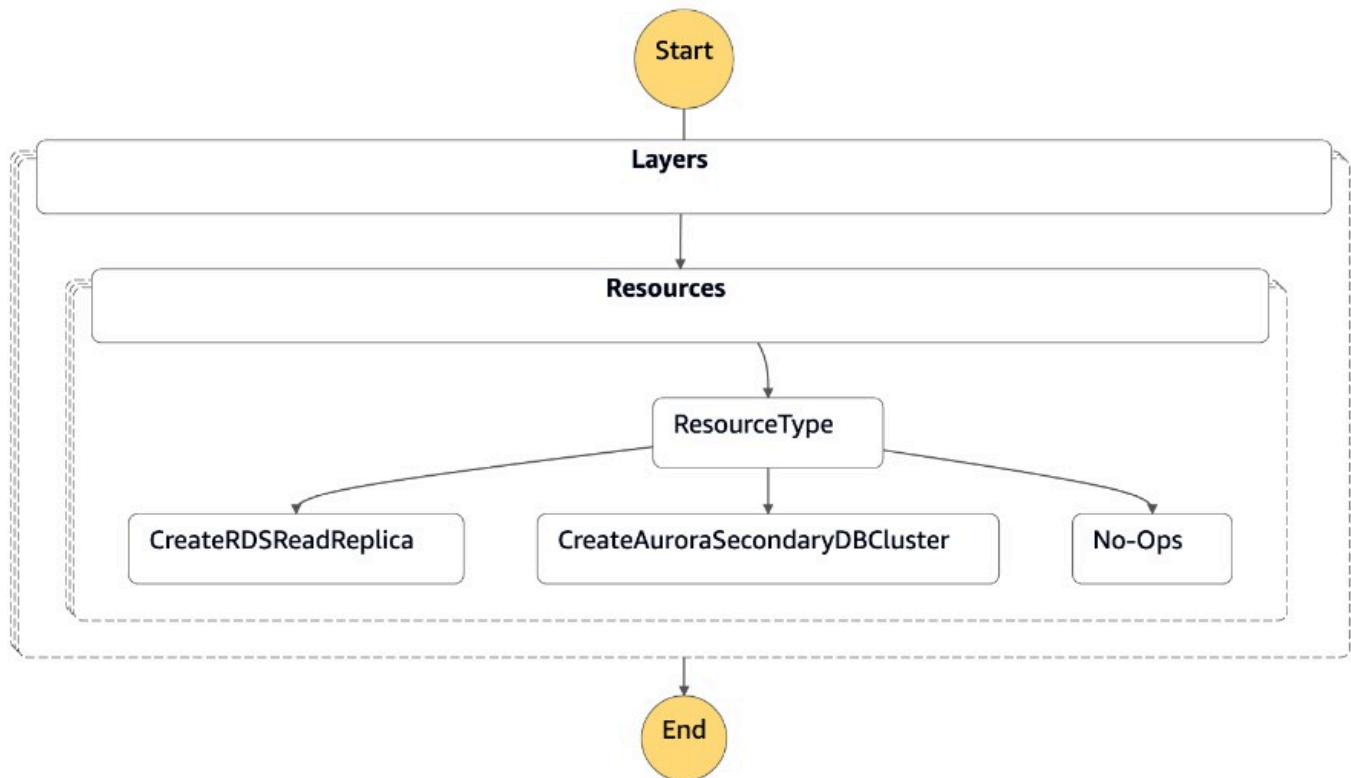
La tabella seguente mostra i parametri utilizzati dalla macchina a DR Orchestrator FAILOVER stati.

Nome del parametro	Description	Valori previsti
layer(obbligatorio: numero)	La sequenza di elaborazione. Tutte le risorse definite nel livello 1 devono essere eseguite prima dell'esecuzione delle risorse del livello 2.	1 o 2 e così via
risorse (richiesto: matrice di dizionari)	Tutte le risorse all'interno di un singolo livello vengono eseguite in parallelo.	<pre> { "resourceType": " String", "resourceName": "St ring", "parameters": { "<param1>": "<!Import cft-output-1">, </pre>

Nome del parametro	Description	Valori previsti
		<pre>.... }</pre>
resourceType (obbligatorio: stringa)	Tipo di risorsa per identificare la risorsa	PromoteRDSReadReplica o FailoverElasticCacheCluster
resourceName (opzionale: stringa)	Per identificare a quale portafoglio di applicazioni appartengono queste risorse	Promote RDS for MySQL Read Replica
parametri (obbligatorio: matrice di dizionario)	Elenco dei parametri necessari per eseguire il failover o il failback del AWS database	<pre>{ "<param1>": "<!Import cft- output-1>", "<param2>": "<!Import cft- output-2>", }</pre>

FAILBACK DI DR Orchestrator

Utilizza la macchina a DR Orchestrator FAILBACK stati dopo l'evento DR, quando la precedente regione primaria () us-east-1 è attiva. Puoi creare la [replica di lettura](#) per Amazon RDS nella precedente regione principale dalla nuova regione principale (us-west-2) per renderla conforme alla tua strategia di disaster recovery. Poiché si tratta di un evento pianificato, puoi pianificare questa attività durante il fine settimana o durante le ore lavorative non di punta con un tempo di inattività stimato.



La macchina a stati accetta parametri in formato JSON, come illustrato nel codice seguente:

```
{
  "StatePayload": [
    {
      "layer": 1,
      "resources": [
        {
          "resourceType": "CreateRDSReadReplica",
          "resourceName": "Create RDS for MySQL Read Replica",
          "parameters": {
            "RDSInstanceIdentifier": "!Import rds-mysql-instance-identifier",
            "TargetClusterIdentifier": "!Import rds-mysql-instance-global-arn",
            "SourceRDSInstanceIdentifier": "!Import rds-mysql-instance-source-identifier",
            "SourceRegion": "!Import rds-mysql-instance-SourceRegion",
            "MultiAZ": "!Import rds-mysql-instance-MultiAZ",
            "DBInstanceClass": "!Import rds-mysql-instance-DBInstanceClass",
            "DBSubnetGroup": "!Import rds-mysql-instance-DBSubnetGroup",
            "DBSecurityGroup": "!Import rds-mysql-instance-DBSecurityGroup",
            "KmsKeyId": "!Import rds-mysql-instance-KmsKeyId",
            "BackupRetentionPeriod": "7",

```

```
        "MonitoringInterval": "60",
        "StorageEncrypted": "True",
        "EnableIAMDatabaseAuthentication": "True",
        "DeletionProtection": "True",
        "CopyTagsToSnapshot": "True",
        "AutoMinorVersionUpgrade": "True",
        "Port": "!Import rds-mysql-instance-DBPortNumber",
        "MonitoringRoleArn": "!Import rds-mysql-instance-RDSMonitoringRole"
    }
}
]
```

Stati Aurora e macchine a stati Step Functions

Questa sezione descrive i processi e le macchine a stati specifici per il failover e il failback dei cluster Amazon Aurora. I cluster sono configurati come database globale.

Note

A scopo dimostrativo, questo esempio utilizza MySQL-Compatible Aurora Edition. È possibile utilizzare passaggi simili per Aurora PostgreSQL-Compatible Edition.

Stato stazionario

Nello stato stazionario, è stato creato un database globale Amazon MySQL-Compatible Aurora `dr-globaldb-cluster-mysql ()` con due cluster DB. Il primo cluster DB (`db-cluster-01`) è stato creato nel sistema primario Regione AWS (`us-east-1`) per servire il carico di lavoro. read/write Il secondo cluster DB (`db-cluster-02`) è stato creato nella regione secondaria (`us-west-2`) per gestire il carico di lavoro di sola lettura.

Oltre a fornire la soluzione DR, è possibile ridurre il carico sul cluster DB primario instradando le query di lettura dalle applicazioni al cluster DB secondario. Ciascuno di questi cluster contiene un'istanza di database denominata `dbcluster-01-use1-instance-1` e `dbcluster-02-usw2-instance-2`, rispettivamente.

Stato dell'evento

Utilizzando un database globale di Amazon Aurora, puoi pianificare e ripristinare i dati in caso di emergenza abbastanza rapidamente. Il ripristino in caso di emergenza viene in genere misurato utilizzando i valori di Recovery Time Objective (RTO) e Recovery Point Objective (RPO). Per ulteriori informazioni, consulta [Utilizzo dello switchover o del failover in un database globale Amazon Aurora](#).

Con un database globale Aurora, esistono due diversi approcci al failover:

- Switchover (failover pianificato gestito)
- Failover (failover manuale non pianificato o scollegamento e promozione)

Passaggio

Lo switchover è destinato ad ambienti controllati, come la manutenzione operativa e altre procedure operative pianificate. Utilizzando un failover pianificato gestito, è possibile trasferire il cluster DB primario del database globale Aurora in una delle regioni secondarie. Poiché lo switchover attende che i cluster DB secondari siano sincronizzati con il database primario, l'RPO è 0 (nessuna perdita di dati). Per ulteriori informazioni, consulta [Esecuzione degli switchover per i database globali di Amazon Aurora](#).

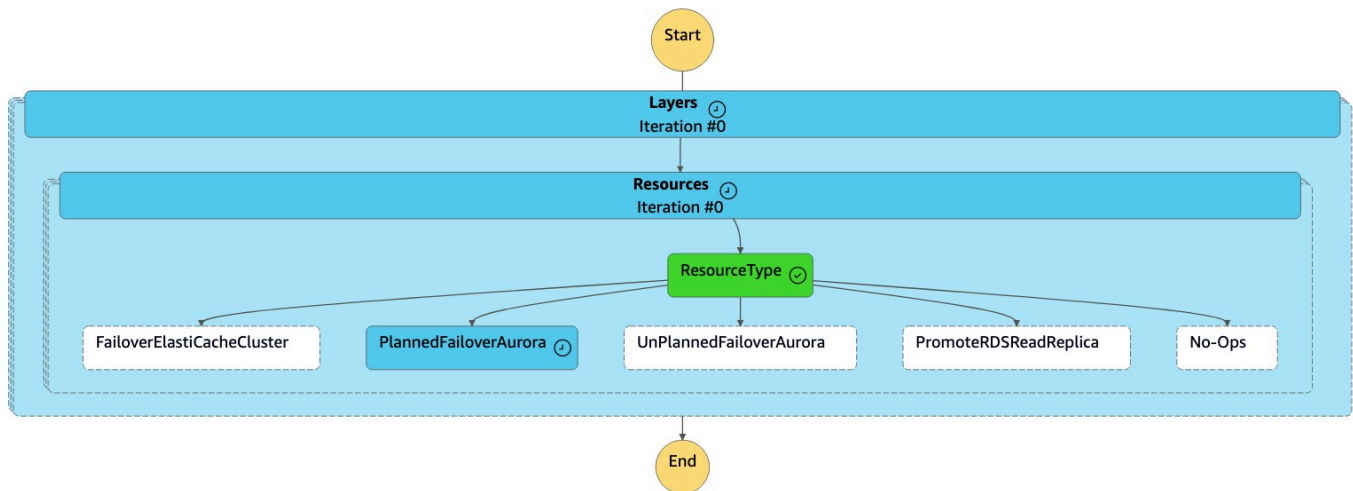
La macchina a `dr-orchestrator-stepfunction-FAILOVER` stati viene richiamata durante lo stato dell'evento per trasferire il cluster primario alla regione secondaria prescelta (`us-west-2`).

Per eseguire lo switchover, effettuate le seguenti operazioni:

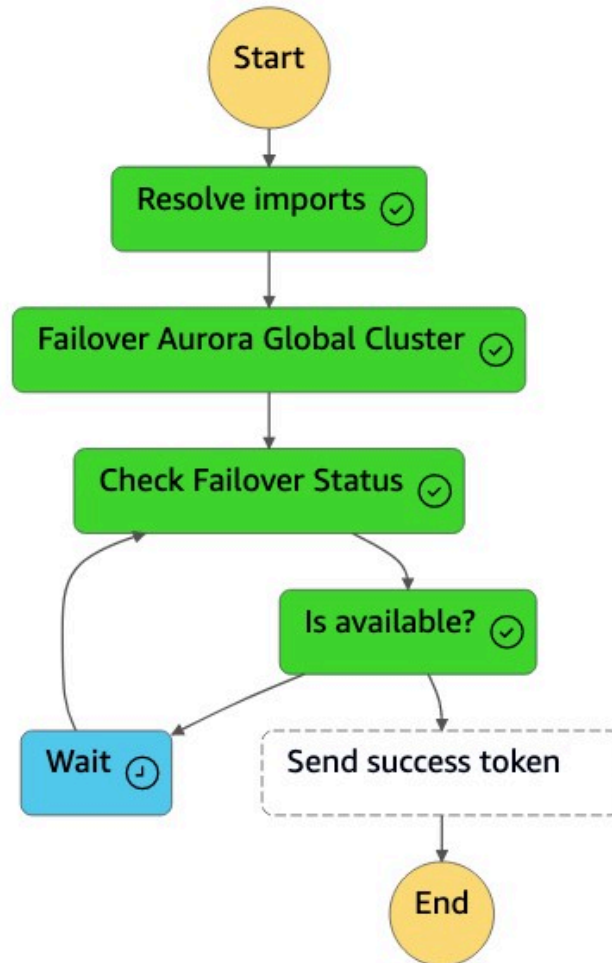
1. Accedi alla Console di gestione AWS.
2. Cambiate la regione nella regione DR (`us-west-2`).
3. Vai a Services e scegli Step Functions.
4. Passa alla macchina a `dr-orchestrator-stepfunction-FAILOVER` stati.
5. Scegli Avvia esecuzione e inserisci il seguente codice JSON nella Input - optional sezione:

```
{
  "StatePayload": [
    {
      "layer": 1,
      "resources": [
        {
          "resourceType": "PlannedFailoverAurora",
          "resourceName": "Switchover (planned failover) of Amazon Aurora global databases (MySQL)",
          "parameters": {
            "GlobalClusterIdentifier": "!Import dr-globaldb-cluster-mysql-global-identifier",
            "DBClusterIdentifier": "!Import dr-globaldb-cluster-mysql-cluster-identifier"
          }
        }
      ]
    }
  ]
}
```

6. La macchina a `dr-orchestrator-stepfunction-FAILOVER` stati legge il tipo di risorsa come `PlannedFailoverAuroraMySQL` e chiama la macchina a `dr-orchestrator-stepfunction-planned-Aurora-failover` stati per eseguire il failover del database globale Aurora.



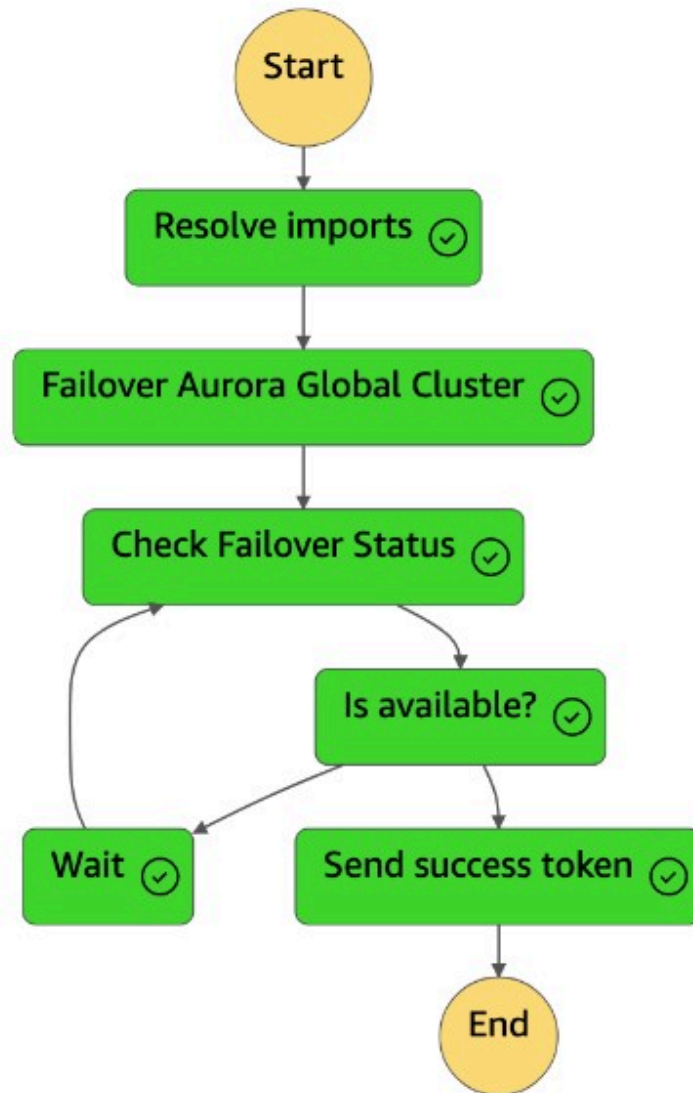
7. La macchina a `dr-orchestrator-stepfunction-planned-Aurora-failover` stati esegue i seguenti passaggi per passare al ruolo del database MySQL-Compatible globale Aurora.



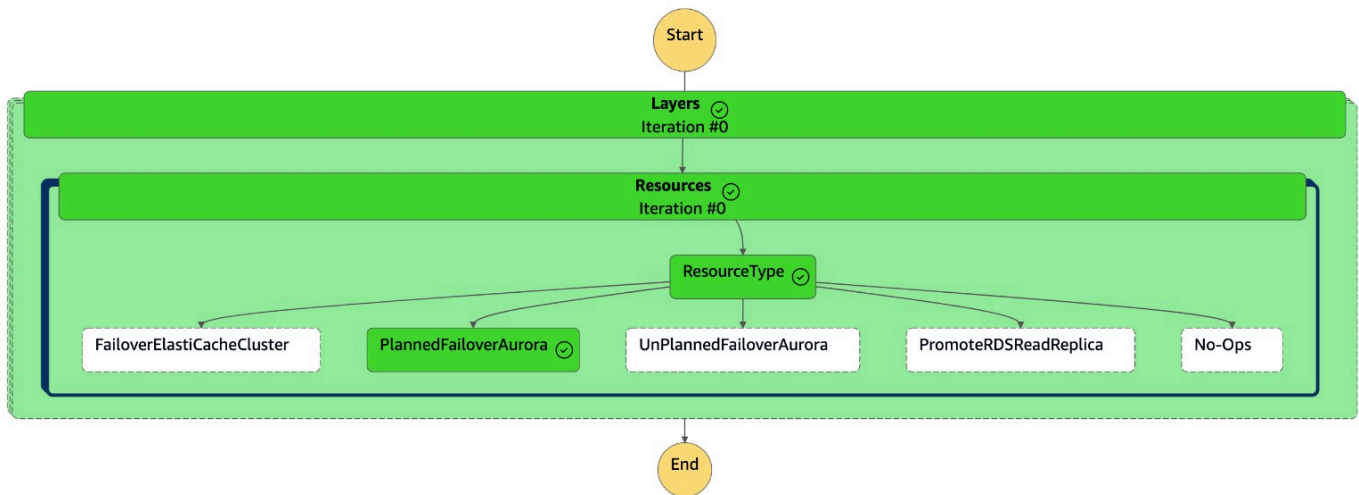
Fase	Description	Valori previsti
Risolvi le importazioni	Una funzione Lambda sostituisce !Import <variable name> i valori con il nome effettivo.	"!Import dr-global db-cluster-mysql-global-identifier" viene sostituito da. "dr-globaldb-cluster-mysql"

Fase	Description	Valori previsti
Failover del cluster globale Aurora	Una funzione Lambda chiama le API failover_global_cluster Boto3 per eseguire il failover del database globale Aurora.	<pre>{ 'GlobalCluster': { 'GlobalClusterId entifier': 'dr-globa ldb-cluster-mysql', 'GlobalClusterRe sourceId': 'cluster- cce7f9bec2846db4', 'GlobalClusterArn': 'arn:aws:rds::xxx', 'Status': 'failing- over', } }</pre>
Verifica lo stato del failover	Una funzione Lambda chiama le API describe_db_clusters Boto3 per verificare lo stato del failover.	modifica, disponibile
Invia token di successo	Una funzione Lambda chiama le API send_task_success Boto3 e invia un token di successo alla macchina a stati. DR Orchestrator Failover	h7XRiCdLtd/83p1E0d MccoxIzFhglsdkzpK9 mBVKZsp7d9yrT1W

- Accedi alla console Amazon RDS. In Status, i valori per il database globale Aurora cambieranno da Disponibile a Commutazione o Modifica.
- Una volta completata, la macchina a `dr-orchestrator-stepfunction-planned-Aurora-failover` stati invia un token di successo alla macchina a `dr-orchestrator-stepfunction-FAILOVER` stati.



10 La macchina a `dr-orchestrator-stepfunction-FAILOVER` stati è completata.



Sulla console, il ruolo del cluster secondario (**dbcluster-02**) è ora cluster primario e il cluster è pronto per servire i carichi di read/write lavoro. Il ruolo del cluster primario originale (dbcluster-01) è ora elencato come Cluster secondario.

Failover manuale non pianificato

In rare occasioni, il database globale Aurora potrebbe subire un'interruzione imprevista del database primario. Regione AWS In questo caso, il cluster di database Aurora primario e il relativo nodo di scrittura non sono disponibili e la replica tra il cluster primario e i secondari cessa. Per ridurre al minimo sia i tempi di inattività (RTO) che la perdita di dati (RPO), lavora rapidamente per eseguire un failover tra regioni e ricostruire il database globale Aurora. Per ulteriori informazioni, consulta [Ripristino di un database globale Amazon Aurora da un'interruzione non pianificata](#).

Per eseguire un failover non pianificato è necessario scollegare il cluster secondario dal database globale Aurora. Prima di eseguire il failover non pianificato, interrompi le scritture delle applicazioni sul cluster Aurora DB primario. Una volta completato correttamente il failover, riconfigura l'applicazione per la scrittura sul nuovo cluster DB primario. Questo approccio aiuta a prevenire la perdita di dati. Inoltre, aiuta a evitare incongruenze nei dati se il nodo di scrittura principale torna online durante il processo di failover.

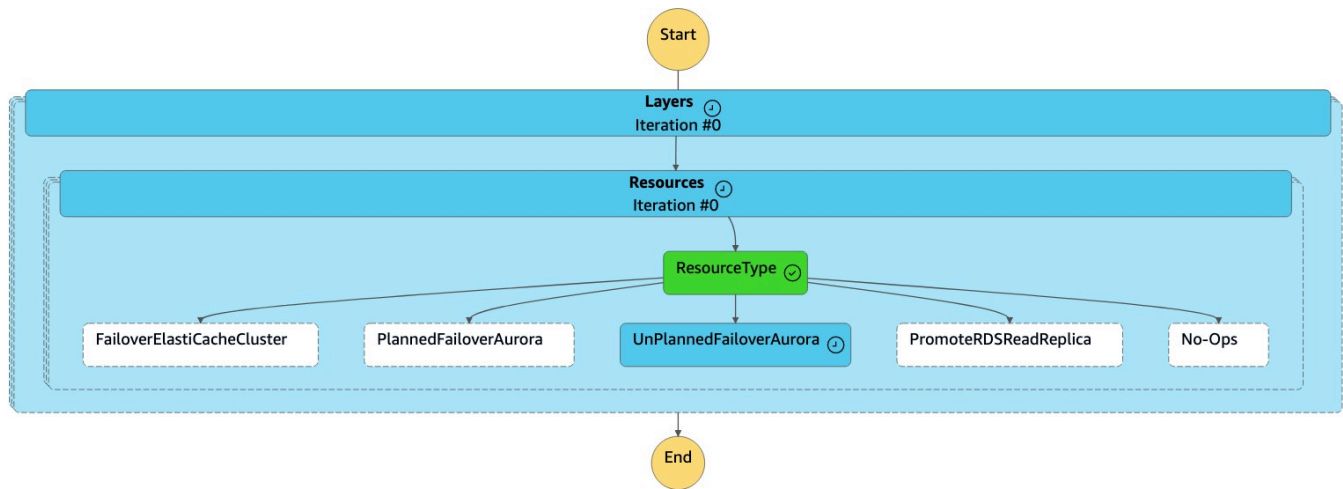
Per eseguire il failover non pianificato, chiamate la macchina a stati. `dr-orchestrator-stepfunction-FAILOVER` In questo esempio, il cluster secondario (db-cluster-02) si trova nella regione DR (us-west-2) in stato stazionario.

Per eseguire il failover, effettuate le seguenti operazioni:

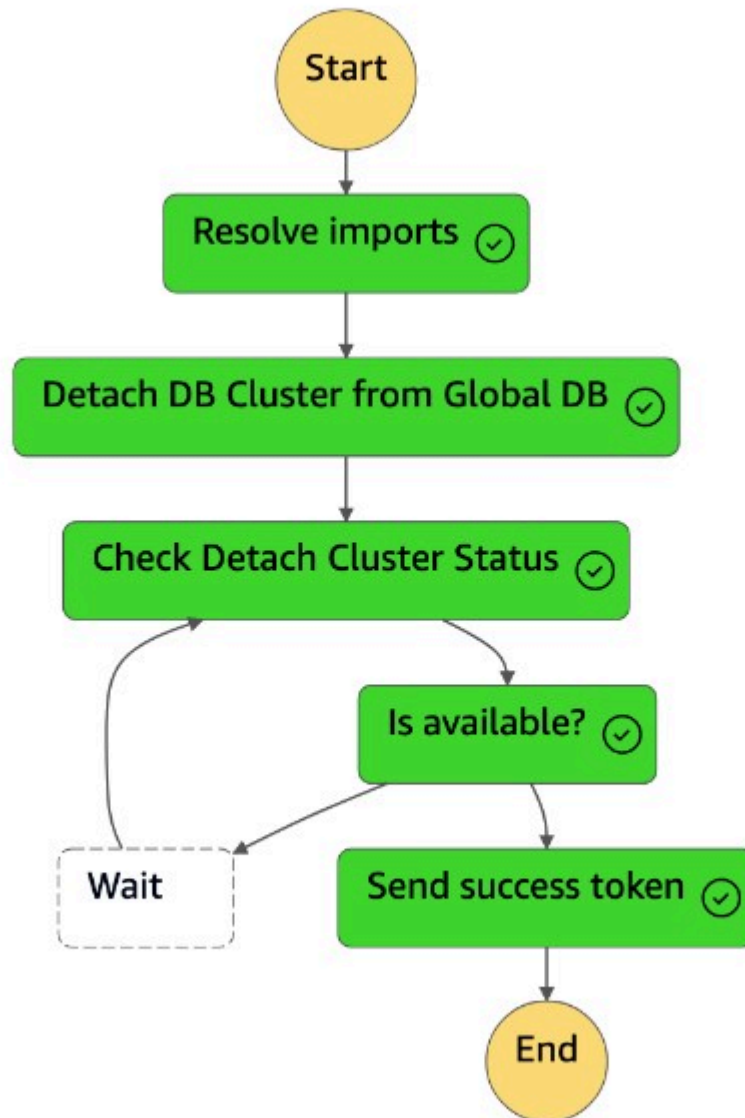
1. Accedi alla console .
2. Cambiate la regione nella regione DR (us-west-2).
3. Vai a Services e scegli Step Functions.
4. Passa alla macchina a `dr-orchestrator-stepfunction-FAILOVER` stati.
5. Scegli Avvia esecuzione e inserisci il seguente codice JSON nella *Input - optional* sezione, usando *UnPlannedFailoverAurora* come: *resourceType*

```
{
  "StatePayload": [
    {
      "layer": 1,
      "resources": [
        {
          "resourceType": "UnPlannedFailoverAurora",
          "resourceName": "Performing unplanned failover for Amazon Aurora global
databases (MySQL)",
          "parameters": {
            "GlobalClusterIdentifier": "!Import dr-globaldb-cluster-mysql-global-
identifier",
            "DBClusterIdentifier": "!Import dr-globaldb-cluster-mysql-cluster-
identifier",
            "ClusterRegion": "!Import dr-globaldb-cluster-mysql-cluster-region"
          }
        }
      ]
    }
  ]
}
```

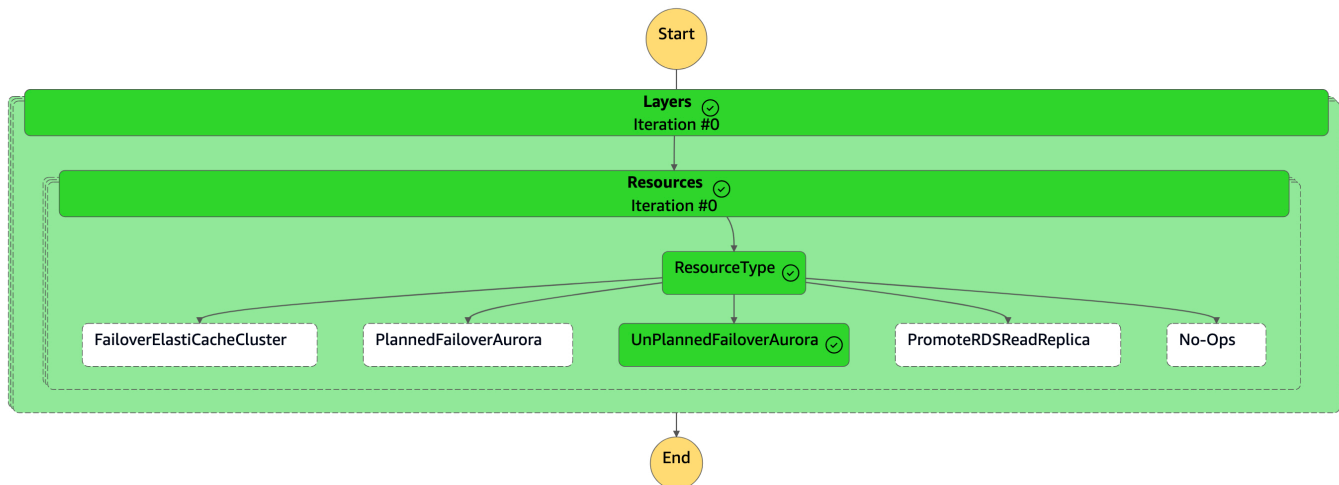
6. La macchina a `dr-orchestrator-stepfunction-FAILOVER` stati legge il tipo di risorsa `UnPlannedFailoverAuroraMySQL` e chiama l'operazione `Detach Cluster from Global Database` dalla macchina a `dr-orchestrator-stepfunction-unplanned-Aurora-failover` stati.



7. L'Detach Cluster from Global Databaseattività scollega (rimuove) il cluster secondario dal database globale.



8. Il cluster secondario (dbcluster-02) viene promosso a diventare un cluster autonomo e può servire carichi di lavoro. read/write
9. La macchina a dr-orchestrator-stepfunction-FAILOVER stati è completata.



10 Il cluster secondario (`dbcluster-02`) viene scollegato dal database globale Aurora e diventa un cluster autonomo per servire il carico di lavoro. read/writer

11 Riconfigura l'applicazione per inviare tutte le operazioni di scrittura a questo nuovo cluster Aurora DB autonomo utilizzando il suo nuovo endpoint del cluster.

Failback

Un failback riporta il database alla posizione principale originale (o nuova) dopo la risoluzione di un disastro (o di un evento pianificato). Una volta risolta l'interruzione non pianificata, potresti voler aggiungere nuovamente la tua regione principale precedente al database globale di Aurora. È necessario innanzitutto eliminare il cluster DB esistente dalla precedente regione primaria, creare un nuovo cluster DB dalla nuova regione primaria e quindi utilizzare il processo di failover pianificato gestito per passare al ruolo del nuovo cluster.

Questa può essere considerata un'attività pianificata che è possibile eseguire durante le ore non di punta o nei fine settimana.

È necessario [modificare manualmente il cluster Amazon Aurora DB](#) e disabilitarlo

DeletionProtection prima di eseguire la macchina a DR Orchestrator FAILBACK stati dalla precedente regione primaria (`us-east-1`) perché è stata creata con. DeletionProtection

DR Orchestrator Framework utilizza la macchina a `dr-orchestrator-stepfunction-FAILBACK` stati per automatizzare i passaggi per eliminare il cluster esistente e creare un nuovo cluster nella precedente regione primaria.

Per disabilitarlo DeletionProtection, procedi come segue:

1. Accedi alla console .
2. Cambia la regione con la precedente regione principale (us-east-1).
3. Accedi alla console Amazon RDS, seleziona il nome del cluster (dbcluster-01) e scegli Modifica.
4. In Protezione da eliminazione, deseleziona la casella di controllo Abilita protezione dall'eliminazione e scegli Continua.
5. Scegli Applica immediatamente, quindi scegli Modifica cluster.

La macchina a DR Orchestrator FAILBACK stati viene richiamata durante il processo di failback dalla precedente regione primaria ()us-east-1.

Per eseguire il failback, effettuate le seguenti operazioni:

1. Accedi alla console .
2. Cambia la regione con la precedente regione principale (us-east-1).
3. Vai a Servizi, quindi scegli Step Functions.
4. Passa alla macchina a DR Orchestrator FAILBACK stati.
5. Scegli Avvia esecuzione e inserisci il seguente codice JSON nella Input - optional sezione:

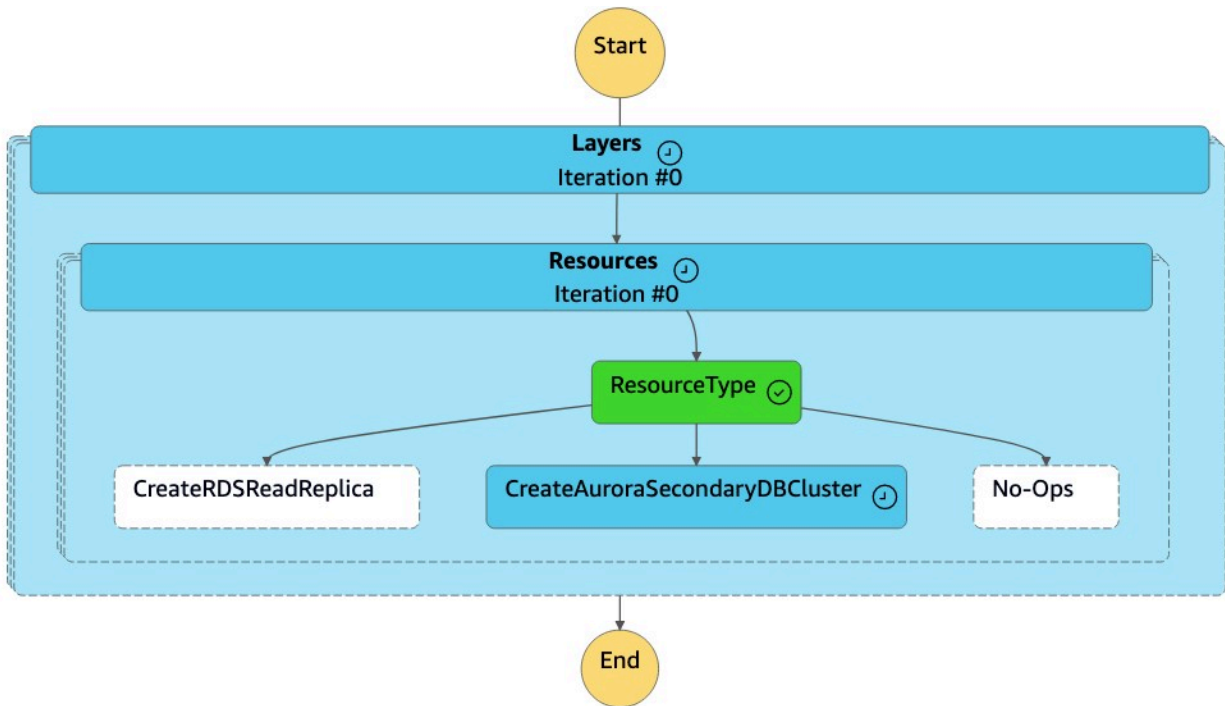
```
{
  "StatePayload": [
    {
      "layer": 1,
      "resources": [
        {
          "resourceType": "CreateAuroraSecondaryDBCluster",
          "resourceName": "To create secondary Aurora MySQL Global Database Cluster",
          "parameters": {
            "GlobalClusterIdentifier": "!Import dr-globaldb-cluster-mysql-global-identifier",
            "DBClusterIdentifier": "!Import dr-globaldb-cluster-mysql-cluster-identifier",
            "DBClusterName": "!Import dr-globaldb-cluster-mysql-cluster-name",
            "SourceDBClusterIdentifier": "!Import dr-globaldb-cluster-mysql-source-cluster-identifier",
            "DBInstanceIdentifier": "!Import dr-globaldb-cluster-mysql-instance-identifier",
            "Port": "!Import dr-globaldb-cluster-mysql-port",
```

```

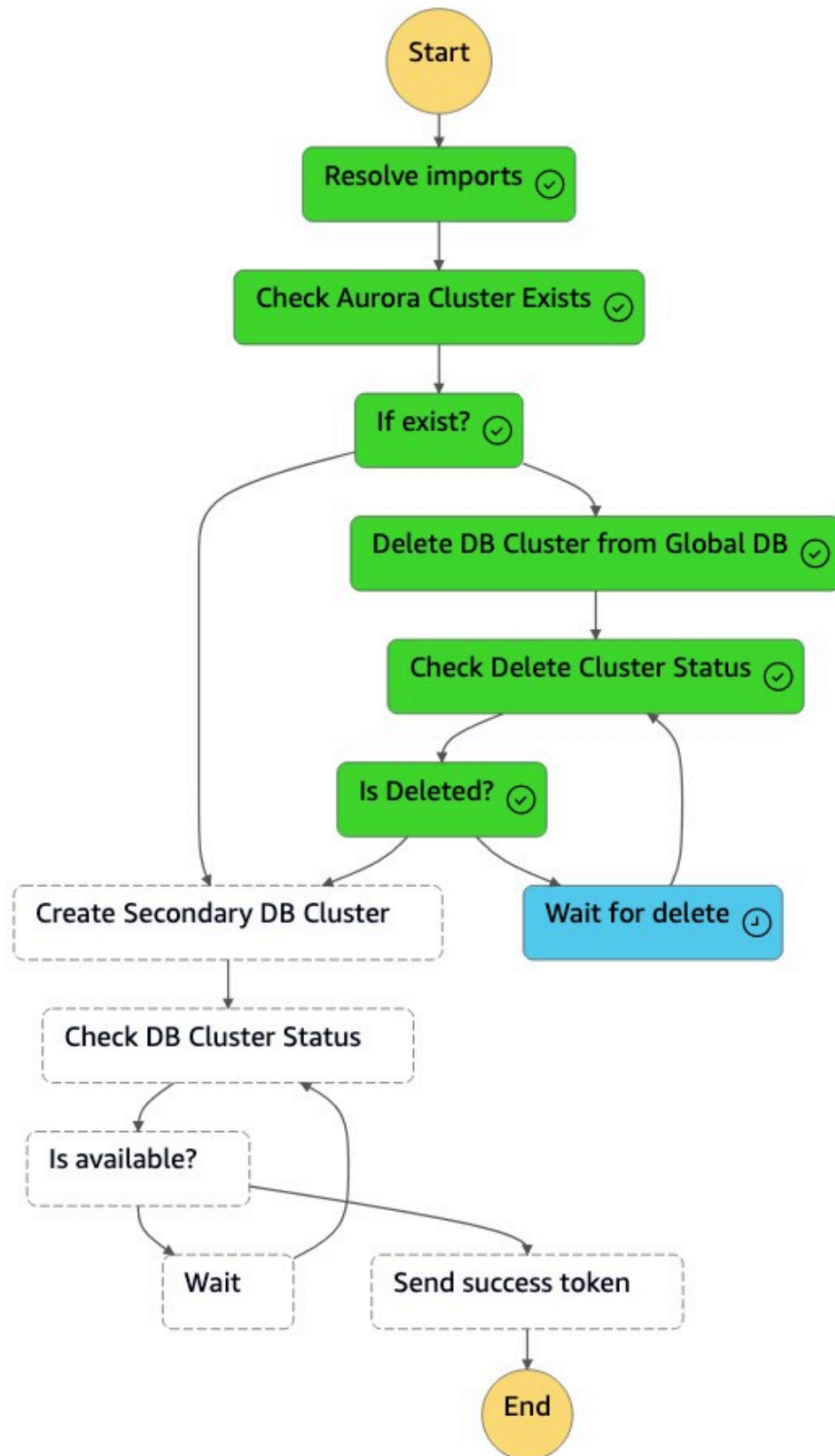
        "DBInstanceClass": "!Import dr-globaldb-cluster-mysql-instance-class",
        "DBSubnetGroupName": "!Import dr-globaldb-cluster-mysql-subnet-group-
name",
        "VpcSecurityGroupIds": "!Import dr-globaldb-cluster-mysql-vpc-security-
group-ids",
        "Engine": "!Import dr-globaldb-cluster-mysql-engine",
        "EngineVersion": "!Import dr-globaldb-cluster-mysql-engine-version",
        "KmsKeyId": "!Import dr-globaldb-cluster-mysql-KmsKeyId",
        "SourceRegion": "!Import dr-globaldb-cluster-mysql-source-region",
        "ClusterRegion": "!Import dr-globaldb-cluster-mysql-cluster-region",
        "BackupRetentionPeriod": "7",
        "MonitoringInterval": "60",
        "StorageEncrypted": "True",
        "EnableIAMDatabaseAuthentication": "True",
        "DeletionProtection": "True",
        "CopyTagsToSnapshot": "True",
        "AutoMinorVersionUpgrade": "True",
        "MonitoringRoleArn": "!Import rds-mysql-instance-RDSMonitoringRole"
    }
}
]
}
]
}

```

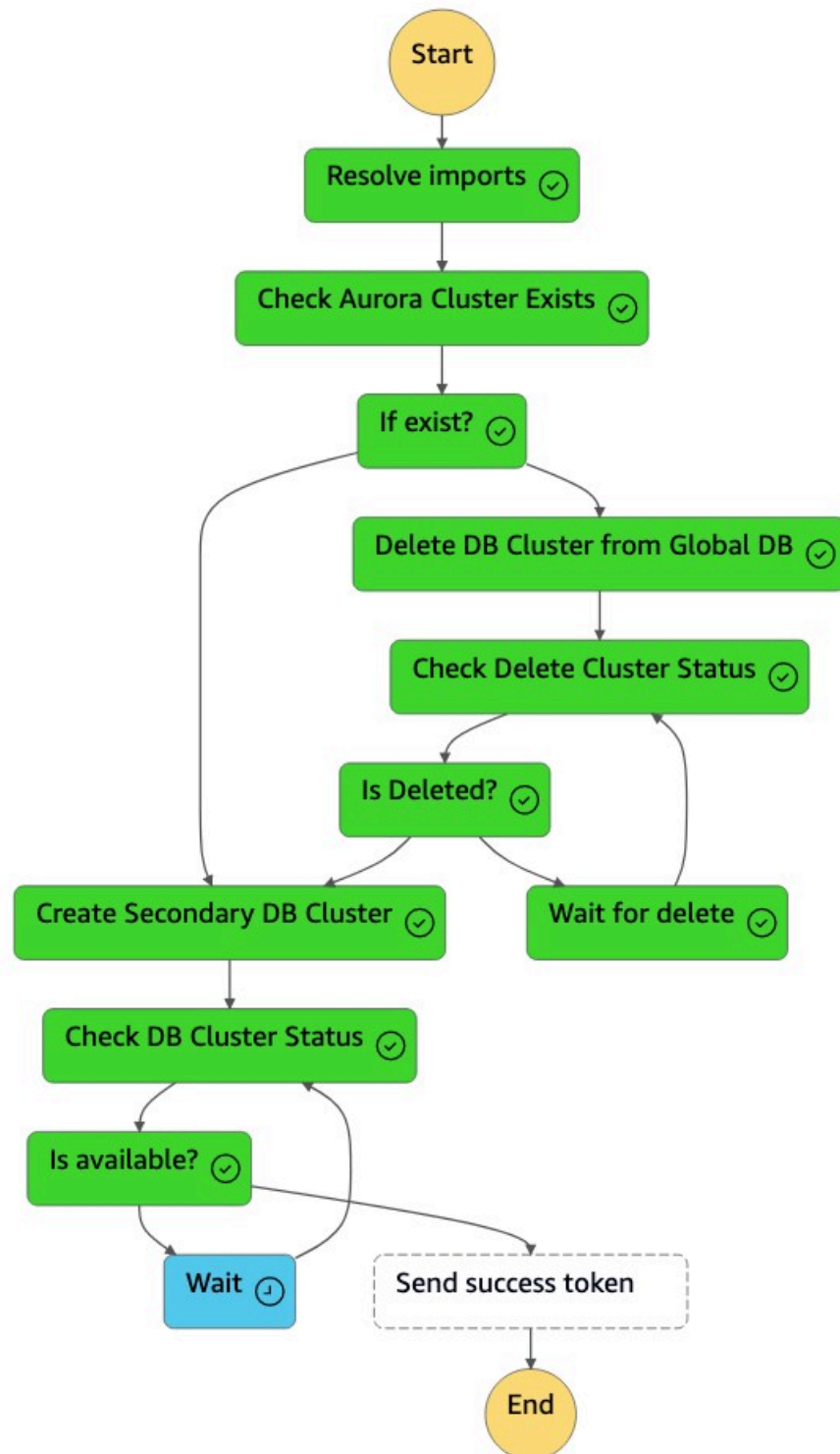
6. La macchina a DR Orchestrator FAILBACK stati legge il tipo di risorsa come `CreateAuroraSecondaryDBCluster` e chiama la macchina a `dr-orchestrator-stepfunction-create-Aurora-Secondary-cluster` stati.



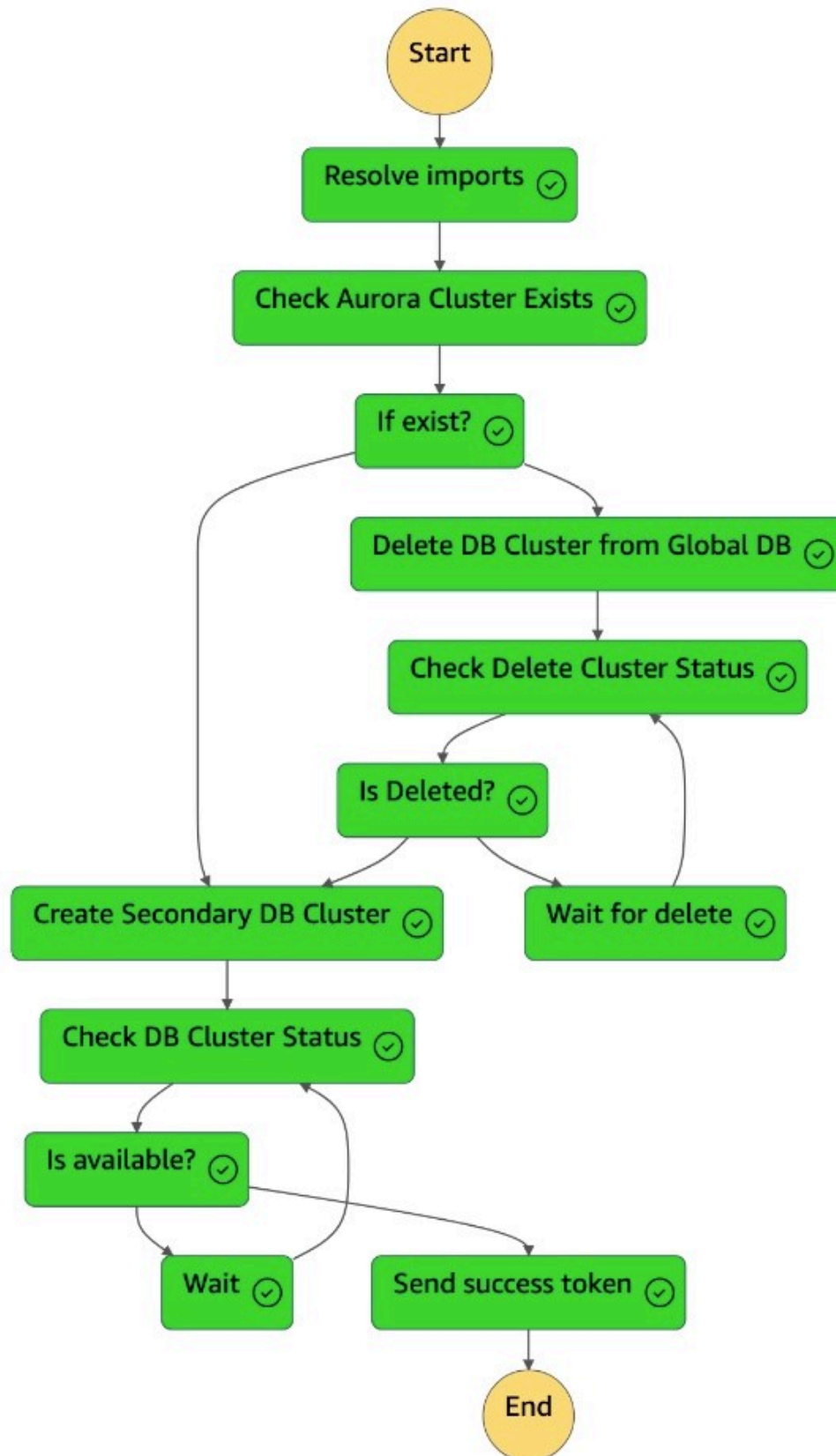
7. La macchina a `dr-orchestrator-stepfunction-create-Aurora-Secondary-cluster` stati elimina il cluster esistente (`dbcluster-01`) dalla precedente regione primaria (`us-east-1`).



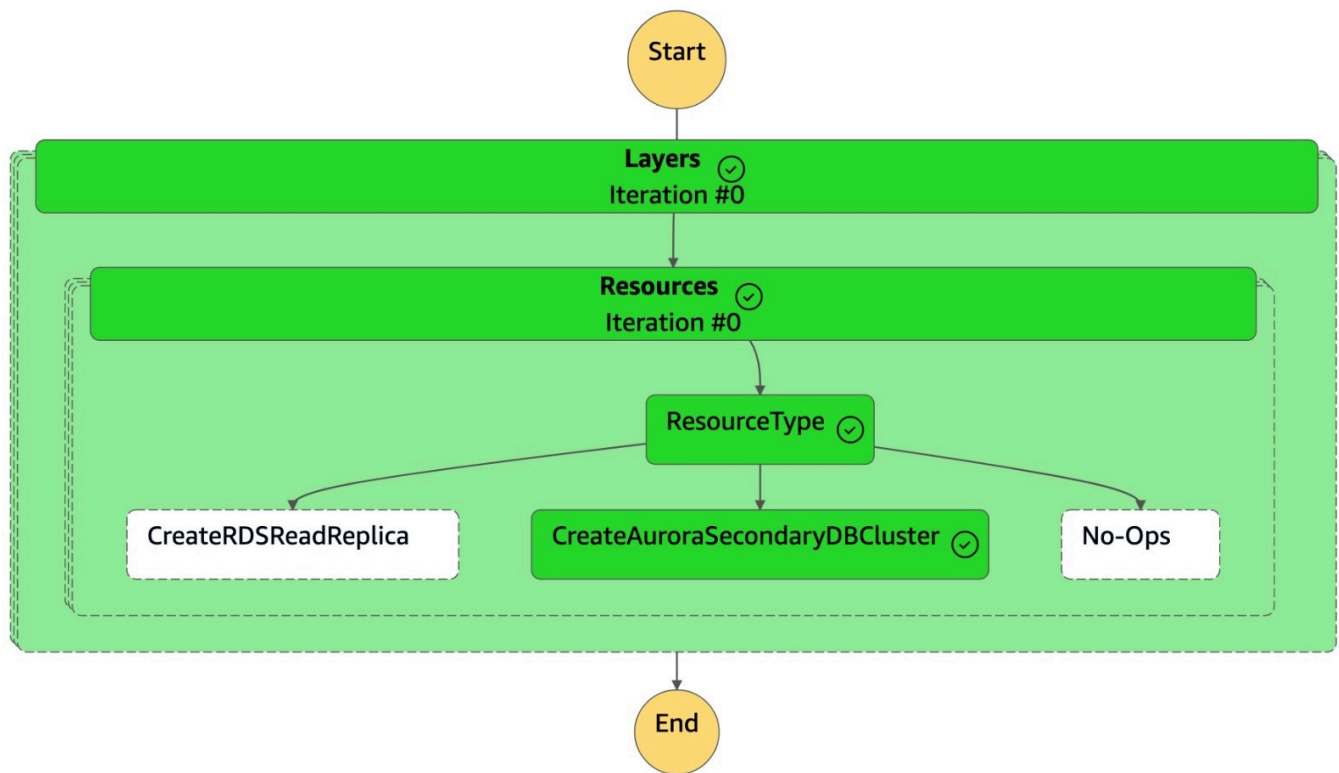
8. Dopo l'eliminazione del cluster (`dbcluster-01`), la macchina a stati crea un nuovo cluster (`dbcluster-01`) insieme all'istanza DB e si unisce al database globale Aurora come cluster secondario per servire carichi di lavoro di sola lettura.



9. Una volta che il cluster secondario è disponibile, la `dr-orchestrator-stepfunction-create-Aurora-Secondary-cluster` macchina a stati viene completata e invia un token di successo alla macchina a DR `Orchestrator Failback` stati.



10 La dr-orchestrator-stepfunction-FAILBACK macchina a stati è completata.



11 Puoi verificare il database globale Aurora sulla console Amazon RDS.

[Se desideri trasferire il cluster DB primario su us-east-1, puoi seguire i passaggi indicati nella sezione Switchover.](#)

Implementa la soluzione

Dopo aver impostato i prerequisiti, segui le istruzioni nel modello [DR Orchestrator Framework](#). Le istruzioni sono incluse per Amazon RDS Aurora e Amazon ElastiCache. Per testare la soluzione, esamina i prerequisiti e le limitazioni e distribuisce database di esempio su AWS.

Prerequisiti

- Un utente attivo [Account AWS](#) per l'implementazione di questa soluzione.
- Autorizzazioni per creare bucket [Amazon Simple Storage Service \(Amazon S3\)](#) e le risorse utilizzate nei modelli. [AWS CloudFormation](#)
- Prima di avviare la macchina a stati: DR Orchestrator FAILBACK
 - [Modifica manualmente l'RDSistanza](#) o [modifica il cluster Amazon Aurora DB](#) e disabilita DeletionProtection
 - Scatta uno [snapshot del DB](#) prima di eliminare l'RDSistanza Amazon o il cluster Aurora se prevedi di eseguire l'analisi della causa principale (). RCA

Limitazioni

- La disponibilità e il supporto delle funzionalità variano a seconda delle versioni specifiche di ciascun motore di database e in tutte le versioni. Regioni AWS Per ulteriori informazioni sulle funzionalità e sulla disponibilità regionale per la replica tra regioni, consulta Repliche di lettura [tra regioni](#).
- Questa soluzione è stata testata utilizzando un'istanza di database principale con una replica di lettura. Se desideri utilizzare più di una replica di lettura, testa accuratamente la soluzione prima di implementarla in un ambiente di produzione.
- Per AmazonRDS, la soluzione supporta solo i seguenti motori: [Amazon RDS for My SQL](#), [Amazon RDS for Postgre e SQL](#) [Amazon for RDS MariaDB](#).

Distribuisce database di esempio su AWS

Per dimostrare la soluzione DR interregionale per database su AWS, puoi distribuire Amazon [RDSfor MySQL](#), [Amazon Aurora SQL My -Compatible Edition](#) e [ElastiCache Amazon \(OSSRedis\)](#) - Global Datastore. [Usa le istruzioni contenute nel file del repository -databases. App-StackREADME aws-cross-region-dr](#) GitHub In questo esempio, la regione principale è us-east-1 la regione secondaria è us-west-2

Lo stack di applicazioni utilizza la sezione AWS CloudFormation [Outputs](#) per esportare i parametri utilizzati da DR Orchestrator Framework per automatizzare il failover e il failback dei database. AWS

Domande frequenti

Trova le risposte alle domande su DR Orchestrator Framework per il failover e il failback.

Quali sono gli RPO e gli RTO che posso ottenere utilizzando questo approccio?

Per informazioni su Recovery Time Objective (RTO) e Recovery Point Objective (RPO), consulta la strategia [Disaster recovery strategy for databases on](#). AWS

È obbligatorio utilizzare le variabili di AWS CloudFormation esportazione?

No, puoi passare direttamente il valore del database globale di Amazon Aurora o dell'istanza database Amazon RDS direttamente nel formato JSON (ad esempio:).

```
"RDSInstanceIdentifier": "rds-mysql-instance"
```

Posso utilizzare il flusso di lavoro FAILOVER di DR Orchestrator per eseguire il failover di più di un database? AWS

Sì, è possibile passare più di una risorsa nel file dei parametri di input per eseguire il failover di più di un database. AWS Il seguente esempio di codice mostra il failover di una replica di lettura Amazon RDS for MySQL e di ElastiCache un datastore globale Amazon (Redis OSS) in parallelo:

```
{
  "StatePayload": [
    {
      "layer": 1,
      "resources": [
        {
          "resourceType": "PromoteRDSReadReplica",
          "resourceName": "Promote RDS MySQL Read Replica",
          "parameters": {
            "RDSInstanceIdentifier": "!Import rds-mysql-instance-identifier",
            "TargetClusterIdentifier": "!Import rds-mysql-instance-global-arn"
          }
        }
      ]
    }
  ]
}
```

```
    },  
    {  
      "resourceType": "FailoverElastiCacheCluster",  
      "resourceName": "Failover ElastiCache Cluster",  
      "parameters": {  
        "GlobalReplicationGroupId": "!Import demo-redis-cluster-global-replication-  
group-id",  
        "TargetRegion": "!Import demo-redis-cluster-target-region",  
        "TargetReplicationGroupId": "!Import demo-redis-cluster-target-replication-  
group-id"  
      }  
    }  
  ]  
}  
]  
}
```

Come posso evitare l' InvalidParameterCombination errore quando eseguo la macchina a stati DR Orchestrator FAILBACK per Amazon RDS?

Il testo completo dell'errore è:

```
"errorMessage": "An error occurred (InvalidParameterCombination) when  
calling the DeleteDBInstance operation: Cannot delete protected DB  
Instance, please disable deletion protection and try again."
```

Per evitare l'errore, [modifica l'istanza RDS](#) disattivandola DeletionProtection prima di avviare la macchina a DR Orchestrator FAILBACK stati.

Passaggi successivi

È possibile utilizzare DR Orchestrator Framework per aggiungere un processo di approvazione prima di avviare le attività di failover o failback. Ad esempio, durante l'avvio del failover durante un evento DR, puoi aggiungere una macchina DR Orchestrator Failover a stati per inviare una [notifica Amazon Simple Notification Service \(Amazon SNS\) tramite](#) e-mail. Non appena viene concessa l'approvazione, l'attività di failover avrà inizio.

Il failover delle applicazioni può anche essere integrato con DR Orchestrator Framework per trasferire le applicazioni di ingresso e punto DNS sul nuovo endpoint dell'istanza o del cluster di database.

È possibile utilizzare la [AWS X-Ray](#) traccia per ottenere la durata del failover per il calcolo dell'RTO. È possibile creare una dashboard di monitoraggio sulla base di X-Ray.

Risorse

- [Replica Regioni AWS tramite datastore globali \(documentazione Amazon ElastiCache \(Redis OSS\)\)](#)
- [Utilizzo del failover in un database globale Amazon Aurora](#) (documentazione di Aurora)
- [Automatizza la gestione degli endpoint di Amazon Aurora Global Database per il failover pianificato e non pianificato](#) (post sul blog)
- [Disaster recovery: 3 scenari di failover per il tuo database globale Amazon Aurora con Terraform \(parte 2\) \(post sul blog\)](#)
- [Strategia di disaster recovery per i database su AWS](#)([strategia](#)AWS Prescriptive Guidance)
- [Automatizza il failover e il failback tra regioni utilizzando DR Orchestrator Framework \(modello Prescriptive Guidance\)](#)AWS

Cronologia dei documenti

La tabella seguente descrive le modifiche significative apportate a questa guida. Per ricevere notifiche sugli aggiornamenti futuri, puoi abbonarti a un [feed RSS](#).

Modifica	Descrizione	Data
Pubblicazione iniziale	—	3 maggio 2024

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.