



Automatice su solución de DR para bases de datos relacionales en AWS

AWS Orientación prescriptiva



AWS Orientación prescriptiva: Automatice su solución de DR para bases de datos relacionales en AWS

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

Las marcas comerciales y la imagen comercial de Amazon no se pueden utilizar en relación con ningún producto o servicio que no sea de Amazon, de ninguna manera que pueda causar confusión entre los clientes y que menosprecie o desacredite a Amazon. Todas las demás marcas registradas que no son propiedad de Amazon son propiedad de sus respectivos propietarios, que pueden o no estar afiliados, conectados o patrocinados por Amazon.

Table of Contents

Introducción	1
Objetivos	2
Descripción general de	3
El proceso de conmutación por error	4
Arquitectura de conmutación por error de DR Orchestrator	4
El proceso de recuperación	5
Arquitectura Failback de DR Orchestrator	5
CONMUTACIÓN POR ERROR DE DR Orchestrator	7
Detalles de los parámetros	8
RECUPERACIÓN POR ERROR DE DR Orchestrator	9
Máquinas de estados Aurora y estados Step Functions	12
Estado estacionario	12
Estado del evento	12
Cambio	13
Conmutación por error manual no planificada	18
Recuperación por recuperación	22
Implementación de la solución	31
Preguntas frecuentes	33
¿Cuáles son el RPO y el RTO que puedo lograr con este enfoque?	33
¿Es obligatorio utilizar variables de AWS CloudFormation exportación?	33
¿Puedo usar el flujo de trabajo de conmutación por error de DR Orchestrator para realizar la conmutación por error de más de una base de datos? AWS	33
¿Cómo puedo evitar el InvalidParameterCombination error al ejecutar la máquina de estados FAILBACK de DR Orchestrator para Amazon RDS?	34
Siguientes pasos	35
Recursos	36
Historial de documentos	37
.....	xxxviii

Automatice su solución de DR para bases de datos relacionales en AWS

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

Mayo de [2024](#) (historia del documento)

Al planificar una [estrategia de recuperación ante desastres \(DR\)](#) en Amazon Web Services (AWS), puede implementar una solución de DR para las bases de datos de su organización. Puede automatizar la solución de recuperación ante desastres configurando una arquitectura basada en eventos para realizar la conmutación por error de las instancias de bases de datos. Entre las ventajas de este enfoque se incluyen las siguientes:

- Reducir los errores humanos causados por la coordinación manual
- Proporcionar el objetivo de tiempo de recuperación (RTO) más corto posible en algunos casos
- Facilitar las pruebas repetibles de su solución de DR

Sin la automatización, una interrupción en su servicio primario Región de AWS requeriría que los recursos de su organización realizaran manualmente los pasos siguiendo un cuaderno de ejecución. Estos pasos pueden incluir varios procesos, como promover réplicas de lectura del Amazon Relational Database Service (Amazon RDS) en la Región de AWS secundaria para que sirvan como nueva instancia de base de datos principal. AWS proporciona un [AWS SDK para Python \(Boto3\)](#) APIs para automatizar estas acciones, que pueden constituir los componentes básicos de una solución de recuperación ante desastres para las necesidades de su organización.

En esta guía se describe un orquestador de DR centralizado y basado en manifiestos que puede ayudarle a conseguir una solución de DR para AWS bases de datos integradas en una misma empresa. Regiones de AWS Cuenta de AWS La decisión de realizar una conmutación por error seguiría necesitando la intervención humana y está fuera del alcance de la automatización.

Los destinatarios de esta guía son arquitectos de aplicaciones, arquitectos de infraestructuras, arquitectos de bases de datos y consultores de bases de datos.

Objetivos

La implementación de la solución DR Orchestrator Framework puede ayudarlo a lograr los siguientes resultados:

- Reduzca los errores manuales durante una interrupción: el rendimiento manual de los procesos de DR puede ser propenso a errores. La automatización de las acciones durante las fases de una actividad de recuperación ante desastres reduce los errores manuales y ayuda a garantizar una conmutación por error o recuperación más fluida.
- Gestione la recuperación ante desastres a gran escala: si su organización tiene una gran flota de bases de datos, la automatización de la recuperación ante desastres le ayudará a ampliar su solución de recuperación ante desastres. Puede automatizar la solución de DR AWS de su base de datos mediante DR Orchestrator Framework. DR Orchestrator automatiza los pasos de conmutación por error o recuperación de las AWS bases de datos mediante un archivo de parámetros en formato JSON.

Descripción general de DR Orchestrator Framework

DR Orchestrator Framework proporciona una solución con un solo clic para organizar y automatizar la recuperación ante desastres entre regiones para bases de datos. AWS Utiliza [AWS Step Functions](#) y lleva [AWS Lambda](#) a cabo los pasos necesarios durante la conmutación por error y la conmutación por recuperación. Las máquinas de estados Step Functions proporcionan la base para la toma de decisiones dentro del diseño del orquestador. Las operaciones de la API para realizar acciones de conmutación por error o conmutación por recuperación se codifican en funciones Lambda que se invocan desde la máquina de estados. Las funciones de Lambda ejecutan [AWS SDK para Python \(Boto3\)](#) API para interactuar con AWS las bases de datos.

DR Orchestrator Framework contiene dos máquinas de estado principales que corresponden a las fases de conmutación por error y conmutación por recuperación.

En Amazon RDS, la fase de conmutación por error convierte una réplica de lectura de RDS entre regiones en una instancia de base de datos independiente. En el caso de Amazon Aurora, cuando la región principal está inactiva durante una interrupción inesperada y poco frecuente, su nodo de grabación no está disponible. La replicación entre el nodo de grabación y los clústeres secundarios se detiene. Debe separar el clúster secundario de la base de datos global y promocionarlo como un clúster independiente. Las aplicaciones pueden conectarse y enviar tráfico de escritura al clúster independiente. Puede utilizar este mismo proceso para [cambiar el clúster de base](#) de datos principal de la base de datos global a las regiones secundarias. Utilice este enfoque para escenarios controlados, como los siguientes:

- Mantenimiento operativo
- Procedimientos operativos planificados
- Promoción de un clúster secundario de Amazon ElastiCache (Redis OSS) como nuevo clúster principal

La fase de recuperación establece la replicación en tiempo real de los datos entre una región principal activa y una nueva región secundaria.

Es fundamental entender que DR Orchestrator solo se aplica a las bases de datos. Es posible que todas las aplicaciones que hacen referencia a estas bases de datos y se encuentran en la misma región necesiten una solución de conmutación por error independiente y en tándem. Tras la conmutación por error de las bases de datos a la región secundaria, es necesario actualizar las

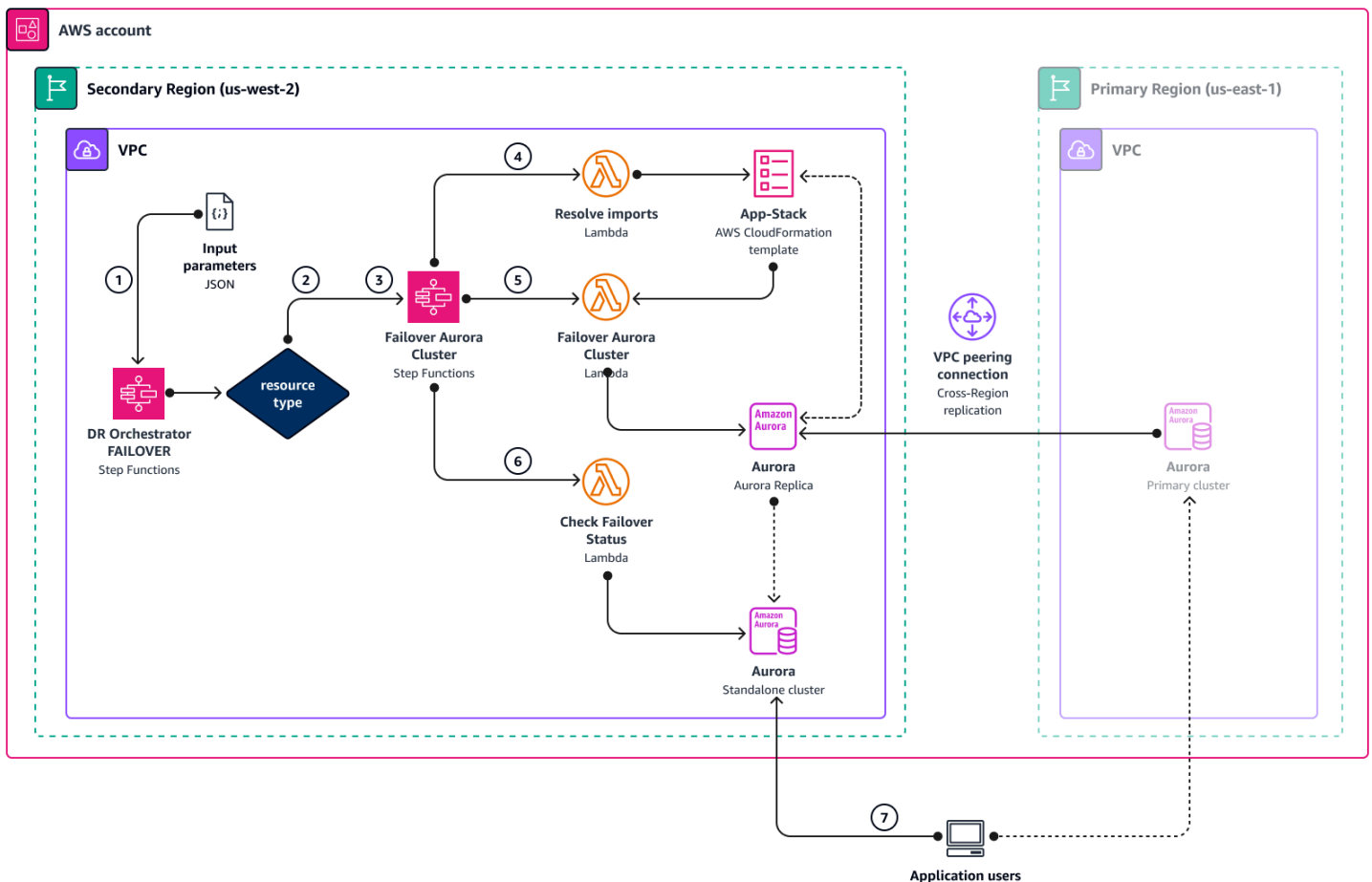
aplicaciones para poder conectarse a las nuevas instancias de bases de datos, que servirán de fuente de datos.

El proceso de conmutación por error

Para realizar una conmutación por error, ejecute la máquina de DR Orchestrator FAILOVER estados. En esta etapa, una base de datos secundaria ya está presente en la región secundaria, ya sea como una réplica de lectura (Amazon RDS) o como un clúster secundario (Amazon Aurora). Al ejecutar la máquina de DR Orchestrator FAILOVER estados, la base de datos secundaria pasa a ser la principal.

Arquitectura de conmutación por error de DR Orchestrator

En el siguiente diagrama, se muestran los conceptos del proceso de conmutación por error para Amazon Aurora cuando se utiliza DR Orchestrator. Amazon Aurora y Amazon ElastiCache utilizan el mismo flujo de trabajo, pero con máquinas de estado y funciones Lambda diferentes.



1. La máquina de DR `Orchestrator FAILOVER` estados lee los parámetros JSON de entrada.
2. Según el `resourceType` parámetro, la máquina de estados llama a otras máquinas de estado: `Promote RDS Read Replica`, `Failover Aurora Cluster`, o `Failover ElastiCache`. Si se pasa más de un recurso en la entrada, estas máquinas de estado se ejecutan en paralelo.
3. La máquina de `Failover Aurora Cluster` estados llama a las funciones Lambda en cada uno de los tres pasos siguientes.
4. La función `Resolve imports Lambda` se resuelve `"! import <export-variable-name>"` con los valores reales de la App-Stack AWS CloudFormation plantilla.
5. La función **Failover Aurora Cluster** Lambda promueve la réplica de lectura como una instancia de base de datos independiente.
6. La función **Check Failover Status** Lambda comprueba el estado de la instancia de base de datos promovida. Cuando el estado es `DISPONIBLE`, la función Lambda envía un token de éxito a la máquina de estado de llamada y lo completa.
7. Puede redirigir las aplicaciones a la base de datos independiente de la región DR (`us-west-2`), que ahora es la base de datos principal.

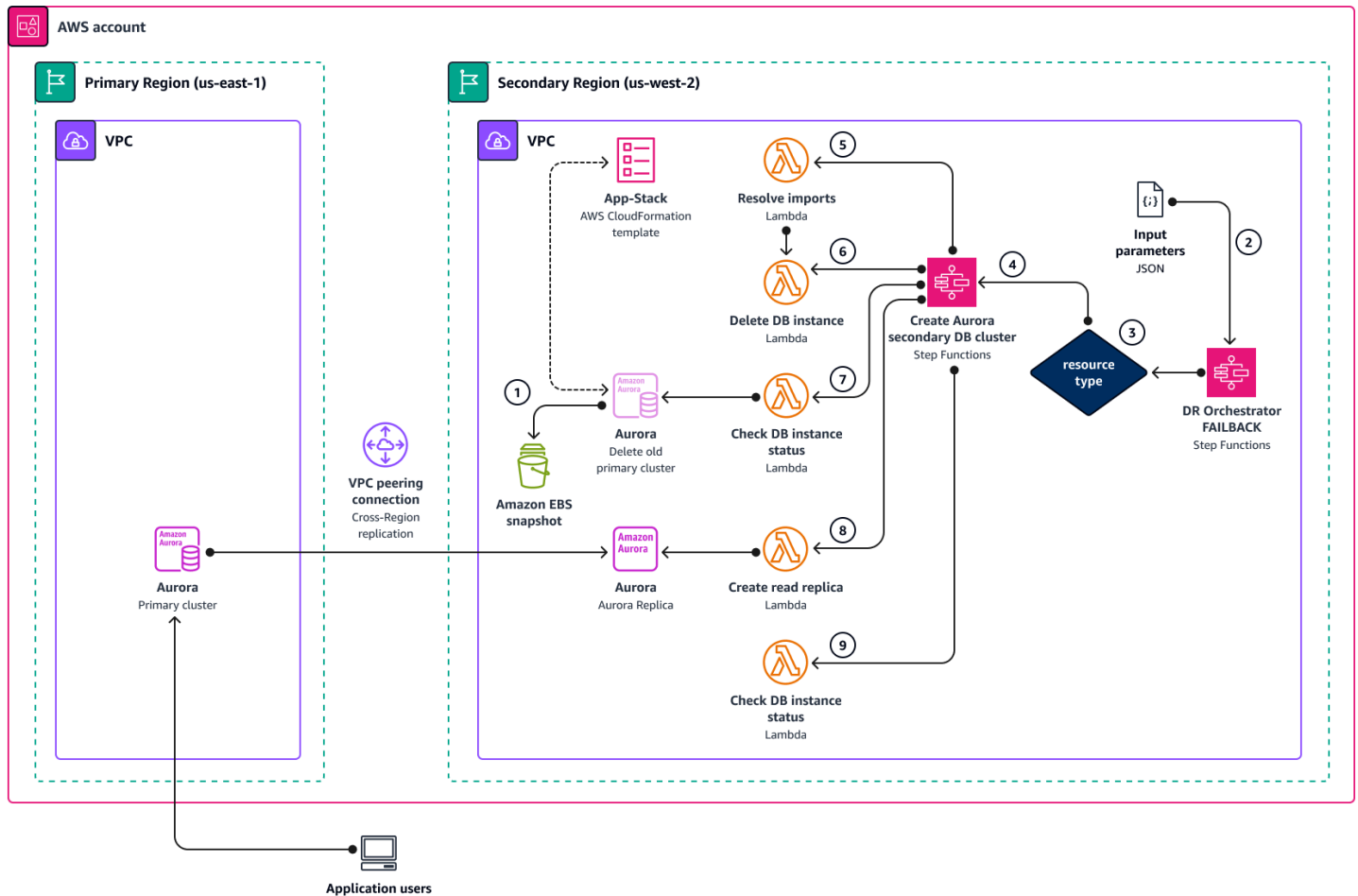
El proceso de recuperación

Cuando la región principal anterior (`us-east-1`) vuelva a estar activa, puede volver a ella por error para que la base de datos `us-east-1` vuelva a ser la principal. Para iniciar la recuperación, ejecute la máquina de DR `Orchestrator FAILBACK` estados. Como su nombre indica, esta máquina de estados comienza a replicar los cambios de la nueva región principal (`us-west-2`) y los devuelve a la antigua región principal (`us-east-1`), que actúa como la secundaria actual.

Una vez establecida la replicación entre las dos regiones, puede iniciar la recuperación por recuperación. Para realizar una recuperación y volver a la región principal original (`us-east-1`), ejecute la máquina de DR `Orchestrator FAILOVER` estados en la región secundaria actual (`us-east-1`) para ascenderla a la región principal.

Arquitectura Failback de DR **Orchestrator**

En el siguiente diagrama, se muestran los conceptos del proceso de recuperación de Amazon Aurora cuando se utiliza DR Orchestrator.



1. Antes de iniciar la recuperación por recuperación, realice una instantánea manual de la base de datos para utilizarla al realizar un análisis de la causa raíz (RCA).

Además, deshabilite la DeletionProtection para el clúster de Aurora en la región principal anterior (us-east-1).

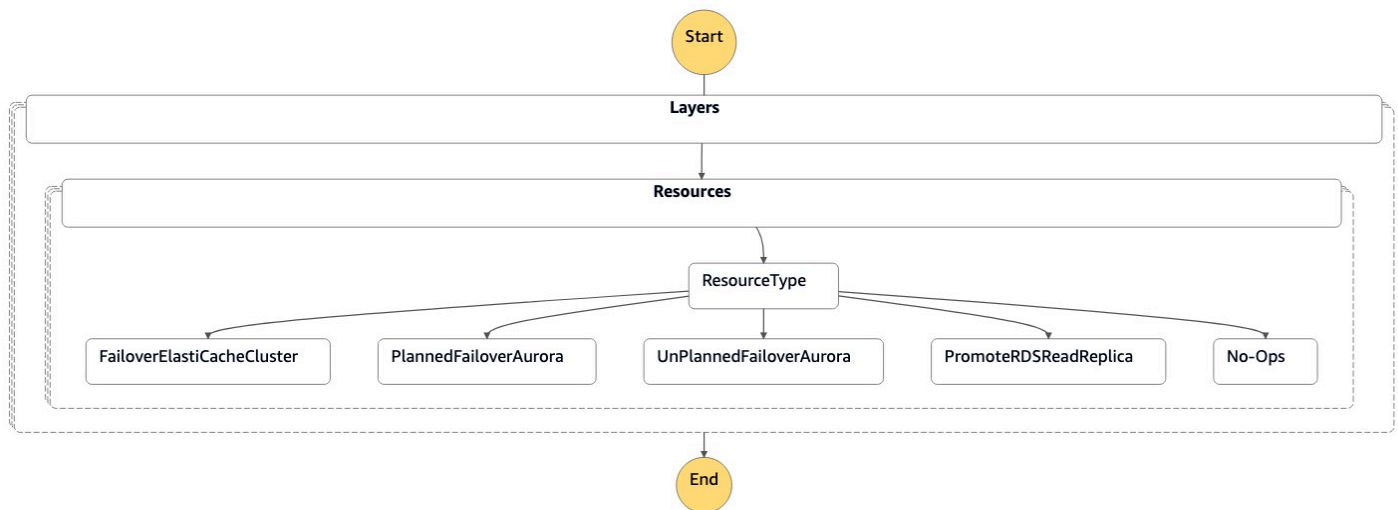
- La máquina de DR Orchestrator FAILBACK estados lee los parámetros JSON de entrada.
- En función de resourceType, la máquina de DR Orchestrator FAILBACK estados llama a la máquina de Create Aurora Secondary DB Cluster estados.
- La máquina de Create Aurora Secondary DB Cluster estados llama a las funciones Lambda en cada uno de los cinco pasos siguientes.
- La función Resolve import Lambda se resuelve "! import <export-variable-name>" con los valores reales de la App-Stack CloudFormation plantilla.
- La función Delete DB Instance Lambda elimina la instancia principal anterior.

7. La función Check DB instance status Lambda comprueba Delete DB Instance status hasta que se borre la base de datos.
8. La función Create Read Replica Lambda crea una réplica de lectura en la región secundaria a partir de la instancia de base de datos que se encuentra en la nueva región principal.
9. La función Check DB instance status Lambda comprueba el estado de la instancia de base de datos de réplica de lectura. Cuando el estado es DISPONIBLE, la función Lambda envía un token de éxito a la máquina de estado que realiza la llamada, la cual se completa.

CONMUTACIÓN POR ERROR DE DR Orchestrator

Utilice la máquina de DR Orchestrator FAILOVER estados en el evento de recuperación ante desastres cuando la región principal (us-east-1) esté inactiva o durante eventos planificados, como el mantenimiento operativo.

Se puede llamar a la función para conmutar por error una o varias bases de datos en paralelo.



La máquina de estados acepta parámetros en formato JSON, como se muestra en el código siguiente:

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

```

Detalles de los parámetros

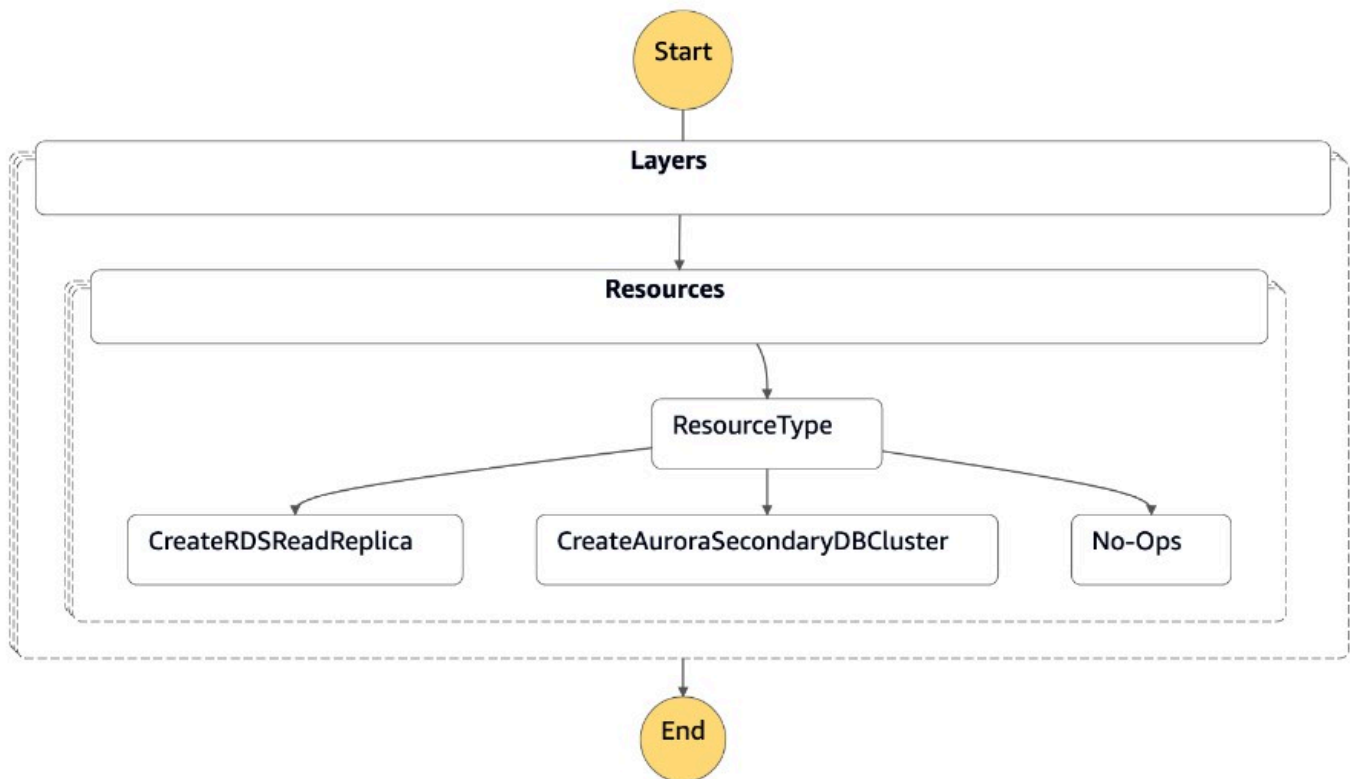
La siguiente tabla muestra los parámetros utilizados por la máquina de DR Orchestrator FAILOVER estados.

Nombre del parámetro	Description (Descripción)	Valores esperados
layer(obligatorio: número)	La secuencia de procesamiento. Todos los recursos definidos en la capa 1 deben ejecutarse antes de ejecutar los recursos de la capa 2.	1 o 2, y así sucesivamente
recursos (obligatorio: conjunto de diccionarios)	Todos los recursos de una sola capa se ejecutan en paralelo.	<pre> { "resourceType": " String", "resourceName": "St ring", </pre>

Nombre del parámetro	Description (Descripción)	Valores esperados
		<pre>"parameters":{ "<param1>":"<!Import cft-output-1">, }</pre>
resourceType (obligatorio: cadena)	Tipo de recurso para identificarlo	PromoteRDSReadReplica o FailoverElastiCacheCluster
resourceName (opcional: cadena)	Para identificar a qué cartera de aplicaciones pertenecen estos recursos	Promote RDS for MySQL Read Replica
parámetros (obligatorio: matriz de diccionarios)	Lista de parámetros necesarios para realizar la conmutación por error o la recuperación de la AWS base de datos	<pre>{ "<param1>": "<!Import cft-output-1">, "<param2>": "<!Import cft-output-2">, }</pre>

RECUPERACIÓN POR ERROR DE DR Orchestrator

Utilice la máquina de DR Orchestrator FAILBACK estados después del evento de DR, cuando la antigua región principal (us-east-1) esté activa. Puede crear la [réplica de lectura](#) de Amazon RDS en la antigua región principal a partir de la nueva región principal (us-west-2) para cumplir con su estrategia de recuperación ante desastres. Como se trata de un evento planificado, puede programar esta actividad durante el fin de semana o durante las horas de menor actividad, con un tiempo de inactividad estimado.



La máquina de estados acepta parámetros en formato JSON, como se muestra en el código siguiente:

```

{
  "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"
  }
}
]
}
]
```

Máquinas de estados Aurora y estados Step Functions

En esta sección se describen las máquinas de proceso y estado específicas de la conmutación por error y la recuperación de clústeres de Amazon Aurora. Los clústeres están configurados como una base de datos global.

Note

Para fines de demostración, en este ejemplo se utiliza Aurora MySQL-Compatible Edition. Puedes usar pasos similares para Aurora PostgreSQL-Compatible Edition.

Estado estacionario

En estado estable, se ha creado una base de datos MySQL-Compatible global de Amazon Aurora (`dr-globaldb-cluster-mysql`) con dos clústeres de bases de datos. El primer clúster de base de datos (`db-cluster-01`) se creó en el clúster principal Región de AWS (`us-east-1`) para atender la read/write carga de trabajo. El segundo clúster de base de datos (`db-cluster-02`) se creó en la región secundaria (`us-west-2`) para alojar la carga de trabajo de solo lectura.

Además de proporcionar la solución de recuperación ante desastres, puede reducir la carga en su clúster de base de datos principal enrutando las consultas de lectura de sus aplicaciones al clúster de base de datos secundario. Cada uno de estos clústeres contiene una instancia de base de datos denominada `dbcluster-01-use1-instance-1` y `dbcluster-02-usw2-instance-2`, respectivamente.

Estado del evento

Al utilizar una base de datos global de Amazon Aurora, puede planificar y recuperarse de un desastre con bastante rapidez. La recuperación tras un desastre normalmente se mide utilizando valores para el objetivo de tiempo de recuperación (RTO) y el objetivo de punto de recuperación (RPO). Para obtener más información, consulte [Uso de la conmutación o la conmutación por error en una base de datos global de Amazon Aurora](#).

Con una base de datos global de Aurora, existen dos enfoques diferentes para la conmutación por error:

- Conmutación (conmutación por error planificada y gestionada)
- Conmutación por error (conmutación por error manual no planificada o desconexión y promoción)

Cambio

La conmutación está destinada a entornos controlados, como el mantenimiento operativo y otros procedimientos operativos planificados. Mediante una conmutación por error planificada y gestionada, puede reubicar el clúster de base de datos principal de su base de datos global de Aurora en una de las regiones secundarias. Como la conmutación espera a que los clústeres de bases de datos secundarios se sincronicen con la base de datos principal, el RPO es 0 (sin pérdida de datos). Para obtener más información, consulte [Realizar cambios en las bases de datos globales de Amazon Aurora](#).

La máquina de `dr-orchestrator-stepfunction-FAILOVER` estados se invoca durante el estado del evento para cambiar el clúster principal a la región secundaria elegida (`us-west-2`).

Para realizar el cambio, haz lo siguiente:

1. Inicie sesión en. Consola de administración de AWS
2. Cambie la región a la región RD (`us-west-2`).
3. Vaya a Servicios y seleccione Step Functions.
4. Navegue hasta la máquina de `dr-orchestrator-stepfunction-FAILOVER` estados.
5. Selecciona Iniciar ejecución e introduce el siguiente código JSON en la Input - optional sección:

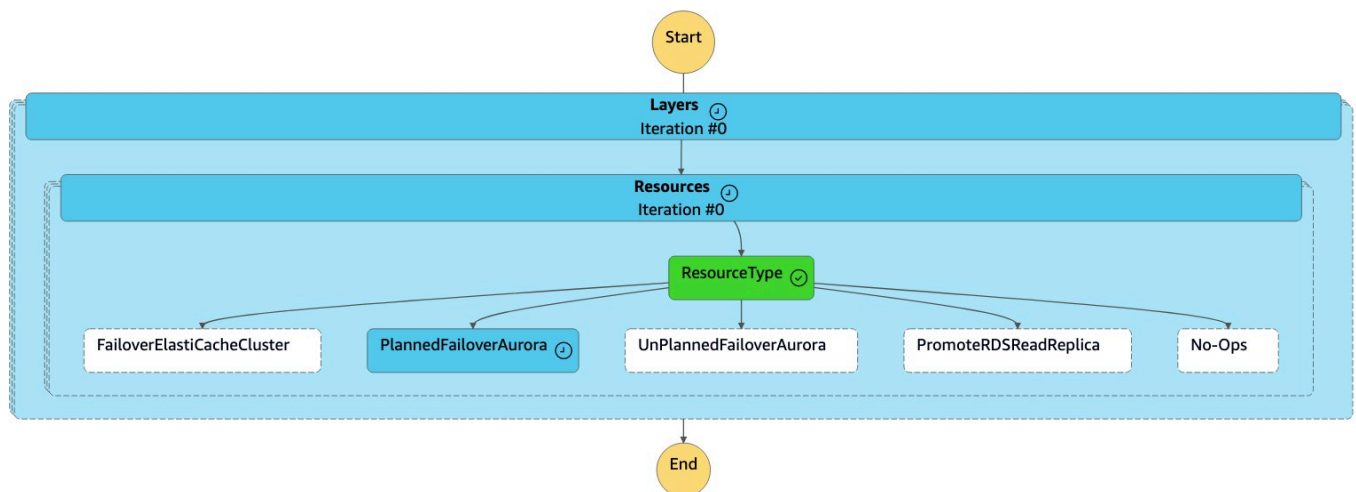
```
{
  "StatePayload": [
    {
      "layer": 1,
      "resources": [
        {
          "resourceType": "PlannedFailoverAurora",
          "resourceName": "Switchover (planned failover) of Amazon Aurora global
databases (MySQL)",
          "parameters": {
            "GlobalClusterIdentifier": "!Import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

```

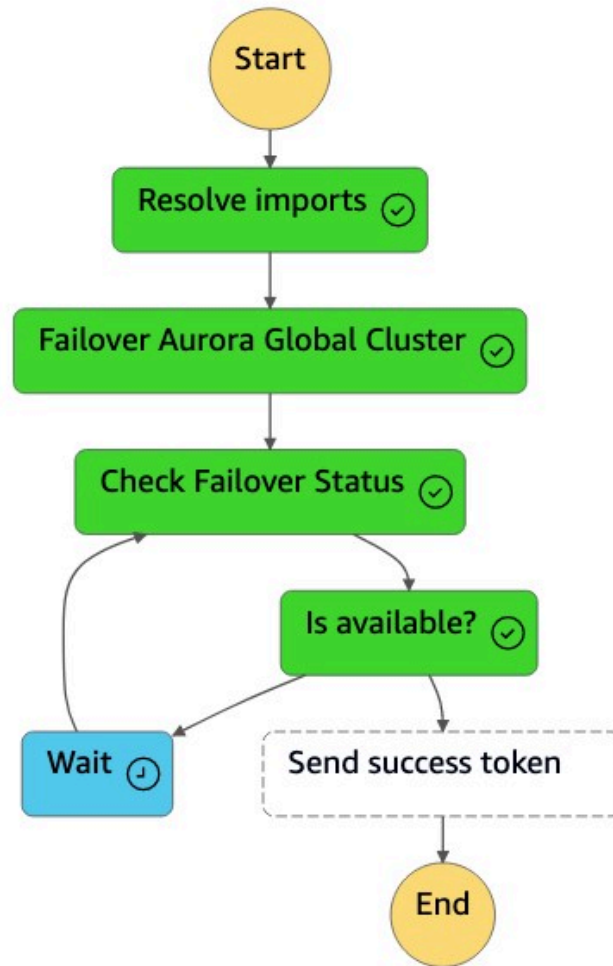
        "DBClusterIdentifier": "!Import dr-globaldb-cluster-mysql-cluster-
identifier"
    }
}
]
}
]
}

```

6. La máquina de `dr-orchestrator-stepfunction-FAILOVER` estados lee el tipo de recurso como `PlannedFailoverAuroraMySQL` y llama a la máquina de `dr-orchestrator-stepfunction-planned-Aurora-failover` estados para conmutar por error la base de datos global de Aurora.



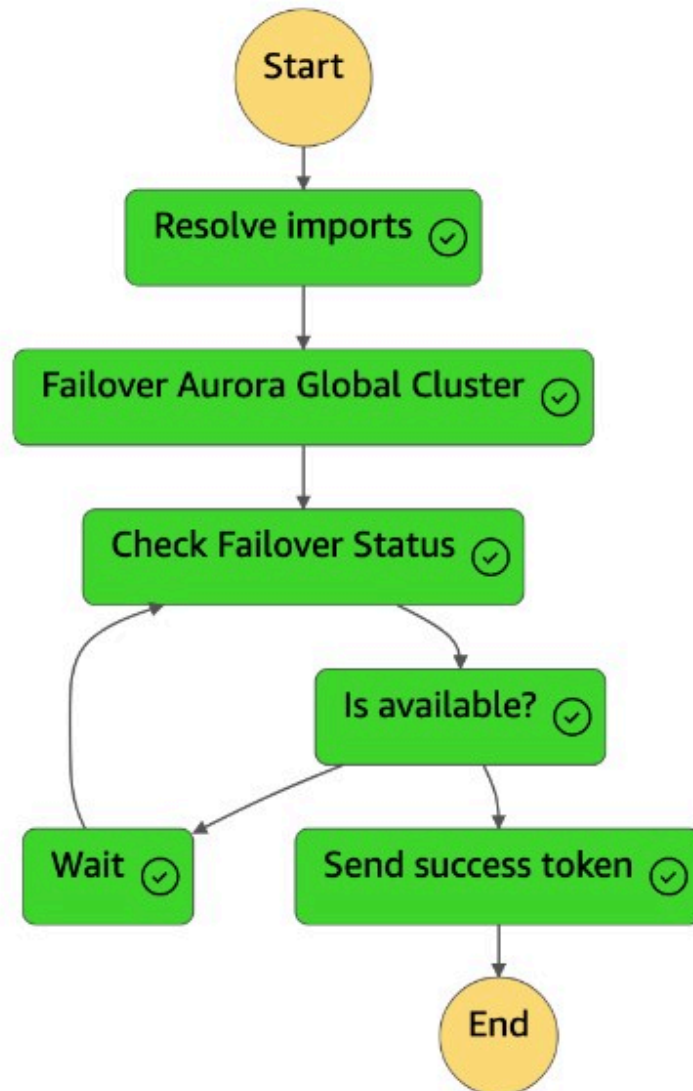
7. La máquina de `dr-orchestrator-stepfunction-planned-Aurora-failover` estados realiza los siguientes pasos para cambiar la función de base de datos MySQL-Compatible global Aurora.



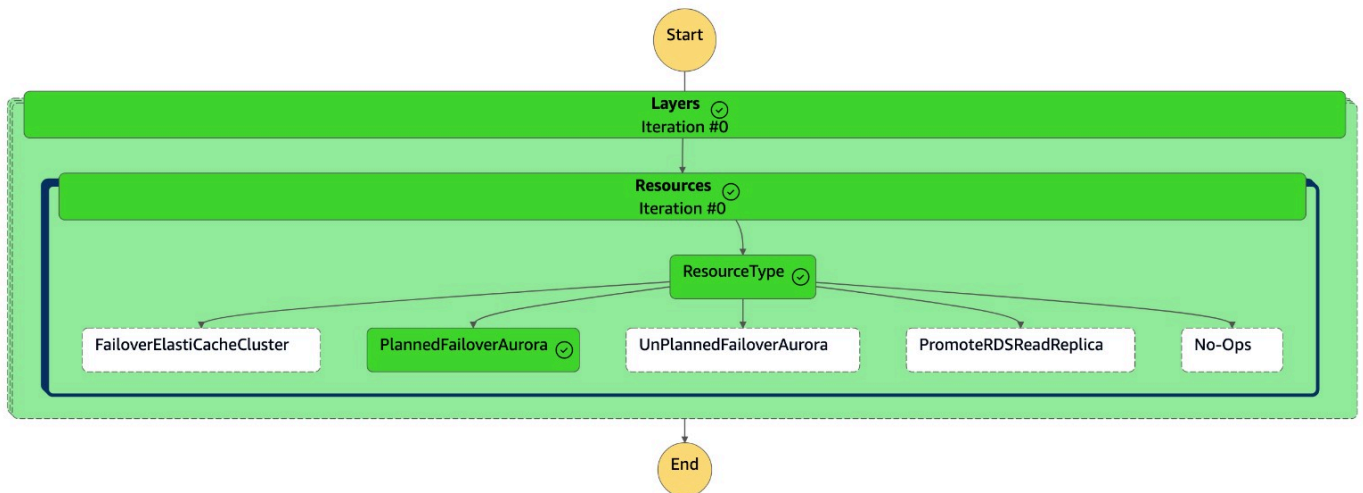
Paso	Description (Descripción)	Valores esperados
Resolver las importaciones	Una función Lambda reemplaza <code>!Import <variable name></code> los valores por el nombre real.	<code>"!Import dr-global db-cluster-mysql-global-identifier"</code> se sustituye por <code>"dr-global db-cluster-mysql"</code> .

Paso	Description (Descripción)	Valores esperados
Clúster global Aurora de conmutación por error	Una función Lambda llama a las API Boto3 failover_global_cluster para realizar la conmutación por error de la base de datos global Aurora.	<pre>{ 'GlobalCluster': { 'GlobalClusterId entifier': 'dr-globa ldb-cluster-mysql', 'GlobalClusterRe sourceId': 'cluster- cce7f9bec2846db4', 'GlobalClusterArn': 'arn:aws:rds::xxx', 'Status': 'failing- over', } }</pre>
Compruebe el estado de la conmutación por error	Una función Lambda llama a las API Boto3 describe_db_clusters para comprobar el estado de la conmutación por error.	modificando, disponible
Envía fichas de éxito	Una función Lambda llama a las API de Boto3 send_task_success y envía un token de éxito a la máquina de estados. DR Orchestrator Failover	h7XRiCdLtd/83p1E0d MccoxIzFhglsdkzpK9 mBVKZsp7d9yrT1W

8. Navegue hasta la consola de Amazon RDS. En Estado, los valores de la base de datos global Aurora cambiarán de Disponible a Conmutación o modificación.
9. Una vez completada la máquina de dr-orchestrator-stepfunction-planned-Aurora-failover estados, envía un testigo de éxito a la máquina de dr-orchestrator-stepfunction-FAILOVER estados.



10 La máquina de `dr-orchestrator-stepfunction-FAILOVER` estados está completa.



En la consola, la función del clúster secundario (`dbcluster-02`) ahora es clúster principal y el clúster está preparado para atender read/write cargas de trabajo. La función del clúster principal original (`dbcluster-01`) ahora aparece como Clúster secundario.

Conmutación por error manual no planificada

En raras ocasiones, la base de datos global de Aurora podría sufrir una interrupción inesperada en su base de datos principal Región de AWS. Si esto sucede, el clúster principal de base de datos de Aurora y su nodo de escritor no estarán disponibles, y cesará la reproducción entre el clúster principal y los secundarios. Para minimizar tanto el tiempo de inactividad (RTO) como la pérdida de datos (RPO), realice rápidamente una conmutación por error entre regiones y reconstruya su base de datos global de Aurora. Para obtener más información, consulte [Recuperación de una base de datos global de Amazon Aurora tras una interrupción imprevista](#).

Para realizar una conmutación por error no planificada, debe separar el clúster secundario de la base de datos global de Aurora. Antes de realizar la conmutación por error no planificada, detenga las escrituras de la aplicación en el clúster de base de datos Aurora principal. Una vez que la conmutación por error se haya completado correctamente, vuelva a configurar la aplicación para que escriba en el nuevo clúster de base de datos principal. Este enfoque ayuda a evitar la pérdida de datos. También ayuda a evitar incoherencias en los datos si el nodo de escritura principal vuelve a estar en línea durante el proceso de conmutación por error.

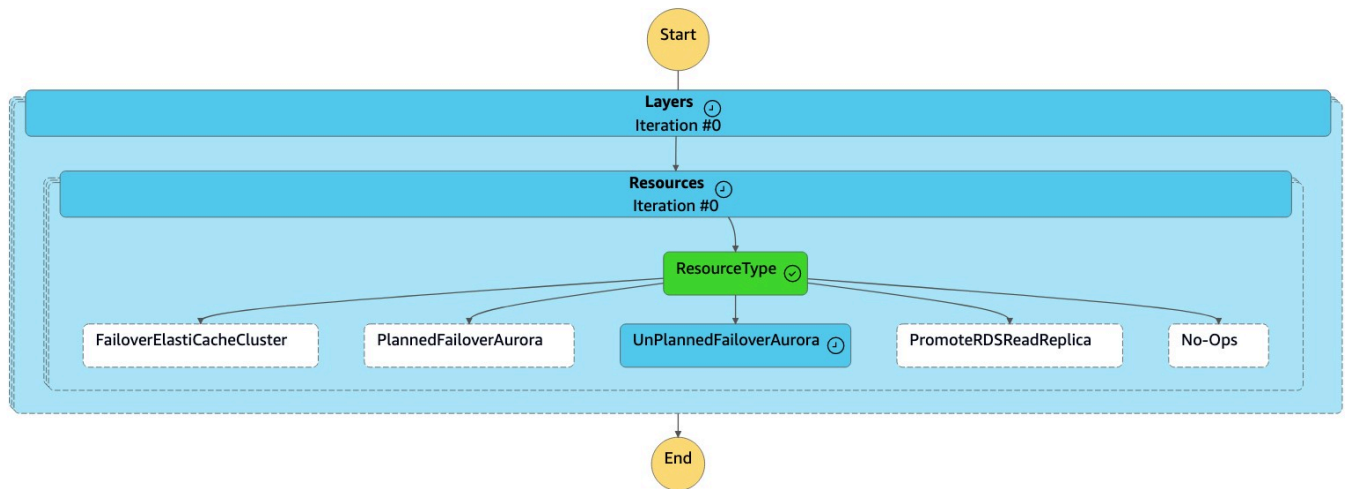
Para realizar la conmutación por error no planificada, llame a la máquina de estado. `dr-orchestrator-stepfunction-FAILOVER` En este ejemplo, el clúster secundario (`dbcluster-02`) se encuentra en la región DR (`us-west-2`) en estado estable.

Para realizar la conmutación por error, haga lo siguiente:

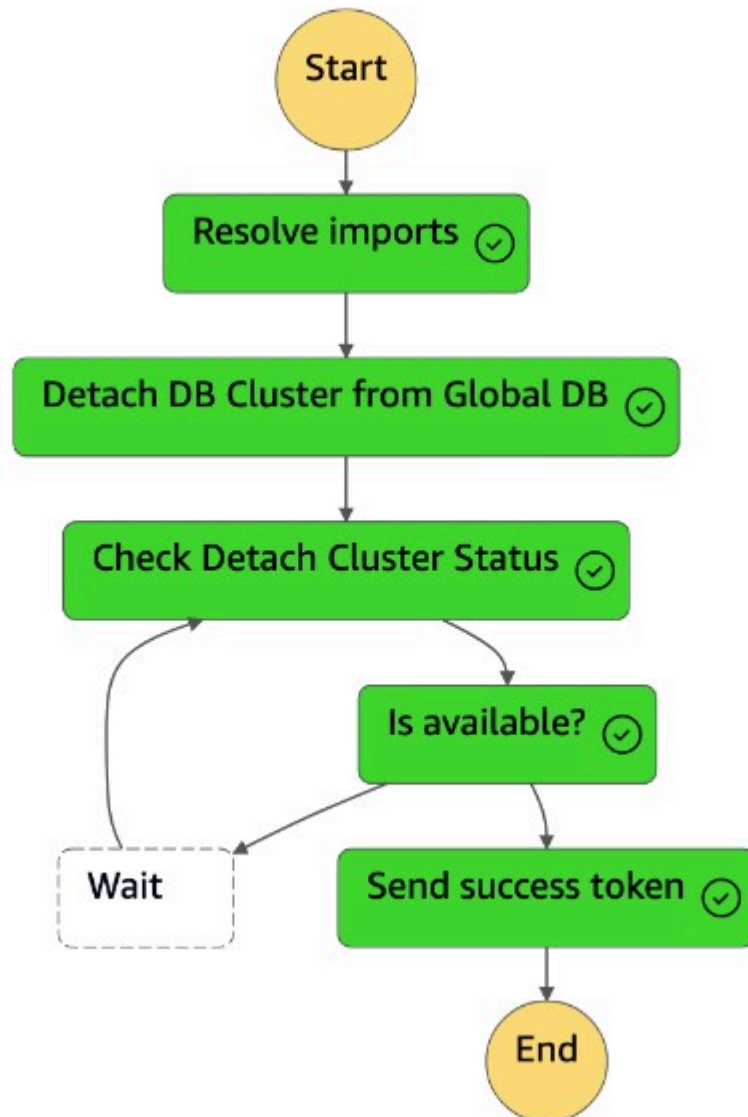
1. Inicie sesión en la consola de .
2. Cambie la región a la región DR (us-west-2).
3. Vaya a Servicios y seleccione Step Functions.
4. Navegue hasta la máquina de `dr-orchestrator-stepfunction-FAILOVER` estados.
5. Seleccione Iniciar ejecución e introduzca el siguiente código JSON en la Input - optional sección, utilizando `UnPlannedFailoverAurora` como `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"
          }
        }
      ]
    }
  ]
}
```

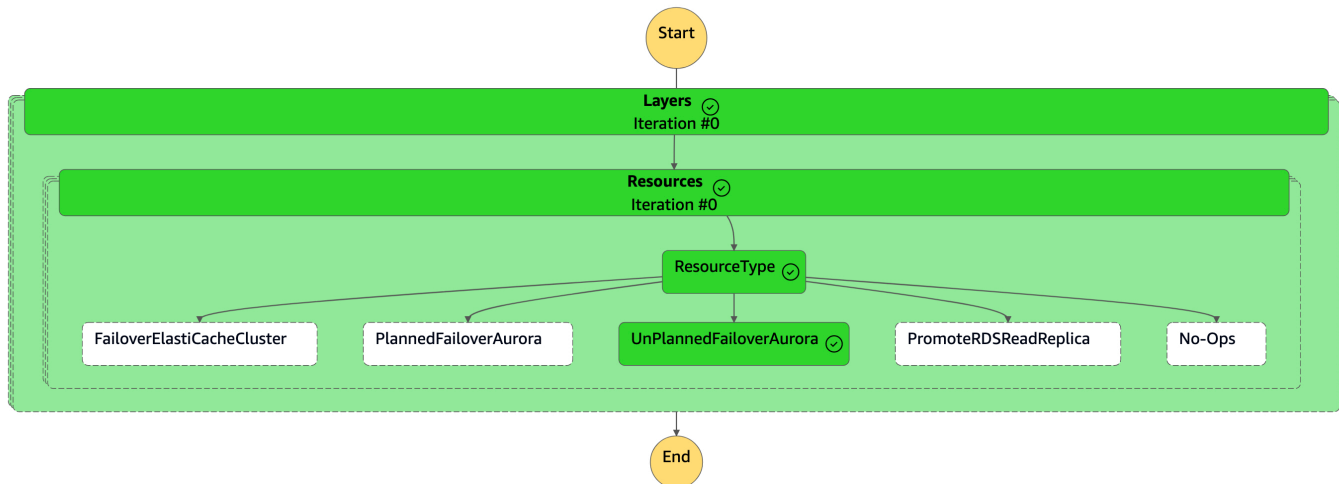
6. La máquina de `dr-orchestrator-stepfunction-FAILOVER` estados lee el tipo de recurso como `UnPlannedFailoverAuroraMySQL` y llama a la tarea `Detach Cluster from Global Database` desde la máquina de `dr-orchestrator-stepfunction-unplanned-Aurora-failover` estados.



7. La Detach Cluster from Global Database tarea separa (elimina) el clúster secundario de la base de datos global.



8. El clúster secundario (dbcluster-02) pasa a ser un clúster independiente y puede atender read/write cargas de trabajo.
9. La máquina dr-orchestrator-stepfunction-FAILOVER de estados está completa.



10 El clúster secundario (`dbcluster-02`) se separa de la base de datos global de Aurora y se convierte en un clúster independiente para atender la read/writer carga de trabajo.

11. Vuelva a configurar la aplicación para enviar todas las operaciones de escritura a este nuevo clúster de base de datos Aurora independiente mediante su nuevo punto final de clúster.

Recuperación por recuperación

Una recuperación por recuperación devuelve la base de datos a la ubicación principal original (o nueva) después de que se resuelva un desastre (o un evento programado). Cuando se haya resuelto la interrupción imprevista, es posible que desee volver a añadir su antigua región principal a la base de datos global de Aurora. Primero debe eliminar el clúster de base de datos existente de la región principal anterior, crear un nuevo clúster de base de datos a partir de la nueva región principal y, a continuación, utilizar el proceso de conmutación por error planificado y gestionado para cambiar la función del nuevo clúster.

Esto puede considerarse una actividad planificada que puede realizar fuera de las horas de mayor actividad o durante un fin de semana.

Debe [modificar manualmente el clúster de base de datos Amazon Aurora](#) e inhabilitarlo `DeletionProtection` antes de ejecutar la máquina de DR Orchestrator `FAILBACK` estado desde la región principal anterior (`us-east-1`) con la que se creó `DeletionProtection`.

DR Orchestrator Framework utiliza la máquina de `dr-orchestrator-stepfunction-FAILBACK` estados para automatizar los pasos necesarios para eliminar el clúster existente y crear uno nuevo en la antigua región principal.

Para deshabilitarlo `DeletionProtection`, haga lo siguiente:

1. Inicie sesión en la consola de .
2. Cambie la región a la antigua región principal (us-east-1).
3. Navegue hasta la consola de Amazon RDS, seleccione el nombre del clúster (dbcluster-01) y elija Modificar.
4. En Protección contra eliminación, desactive la casilla Habilitar la protección contra eliminación y seleccione Continuar.
5. Seleccione Aplicar inmediatamente y, a continuación, seleccione Modificar clúster.

La máquina de DR `Orchestrator FAILBACK` estados se invoca durante el proceso de recuperación desde la región principal anterior (us-east-1).

Para realizar la conmutación por recuperación, haga lo siguiente:

1. Inicie sesión en la consola de .
2. Cambie la región a la antigua región principal (us-east-1).
3. Vaya a Servicios y, a continuación, seleccione Step Functions.
4. Navegue hasta la máquina de DR `Orchestrator FAILBACK` estados.
5. Selecciona Iniciar ejecución e introduce el siguiente código JSON en la Input - optional sección:

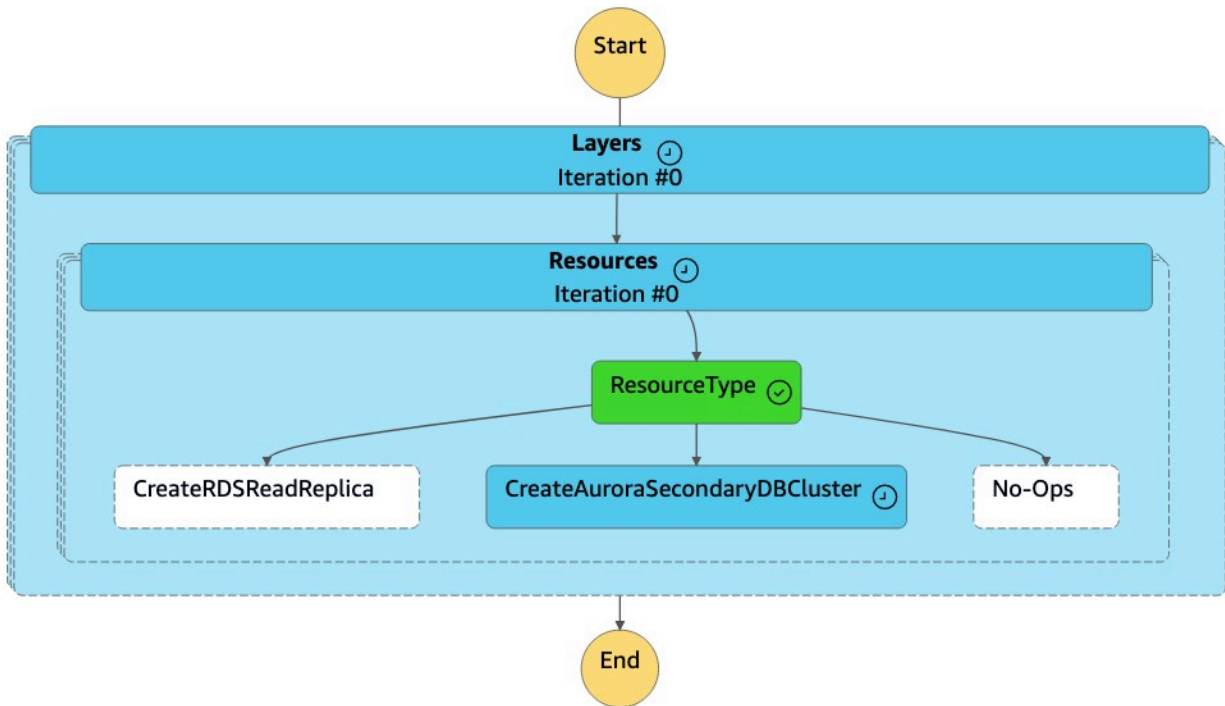
```
{
  "StatePayload": [
    {
      "layer": 1,
      "resources": [
        {
          "resourceType": "CreateAuroraSecondaryDBCluster",
          "resourceName": "To create secondary Aurora MySQL Global Database Cluster",
          "parameters": {
            "GlobalClusterIdentifier": "!Import 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",
```

```

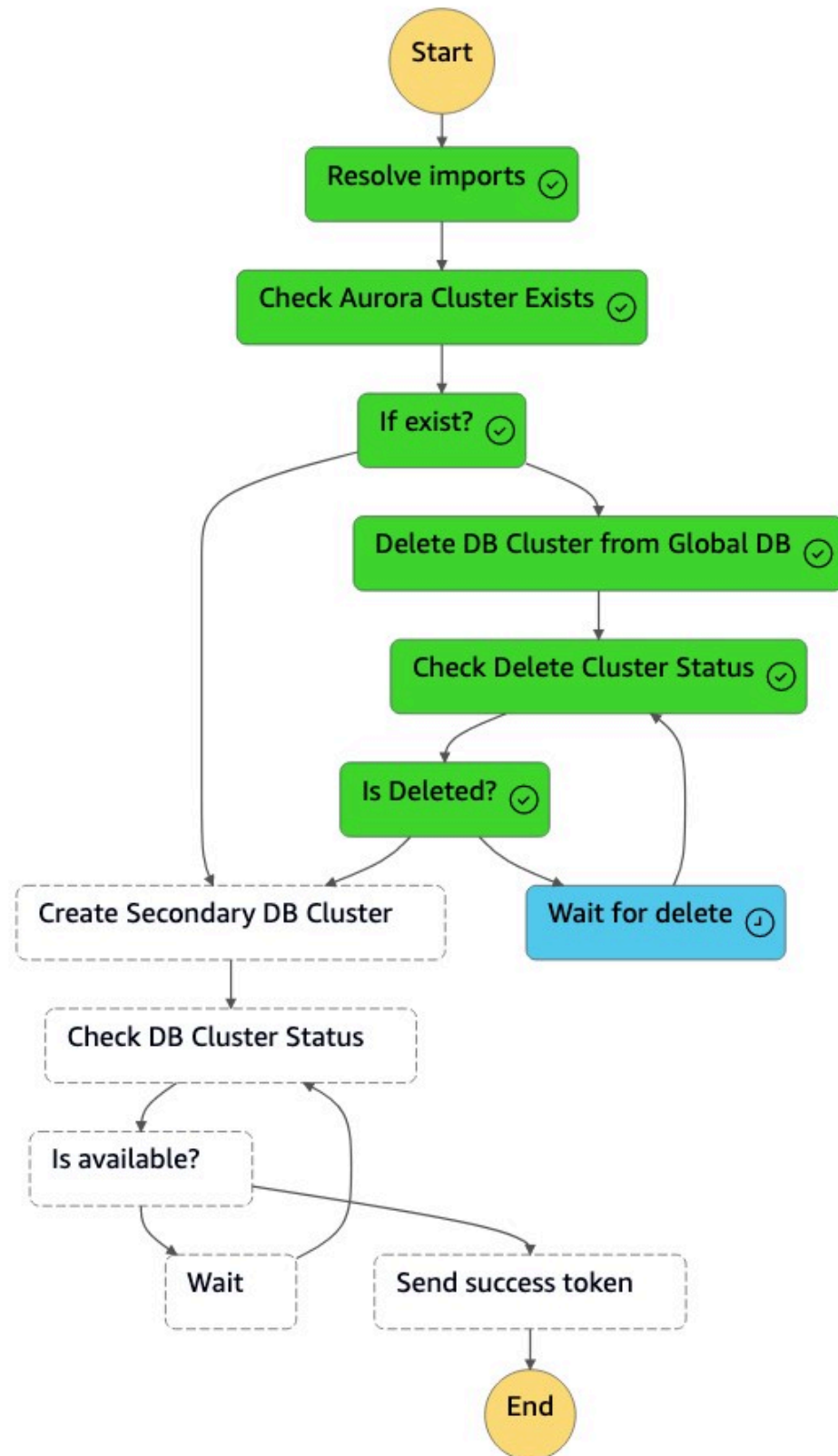
        "DBInstanceIdentifier": "!Import dr-globaldb-cluster-mysql-instance-
identifier",
        "Port": "!Import dr-globaldb-cluster-mysql-port",
        "DBInstanceClass": "!Import dr-globaldb-cluster-mysql-instance-class",
        "DBSubnetGroupName": "!Import dr-globaldb-cluster-mysql-subnet-group-
name",
        "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"
    }
}
]
}
]
}

```

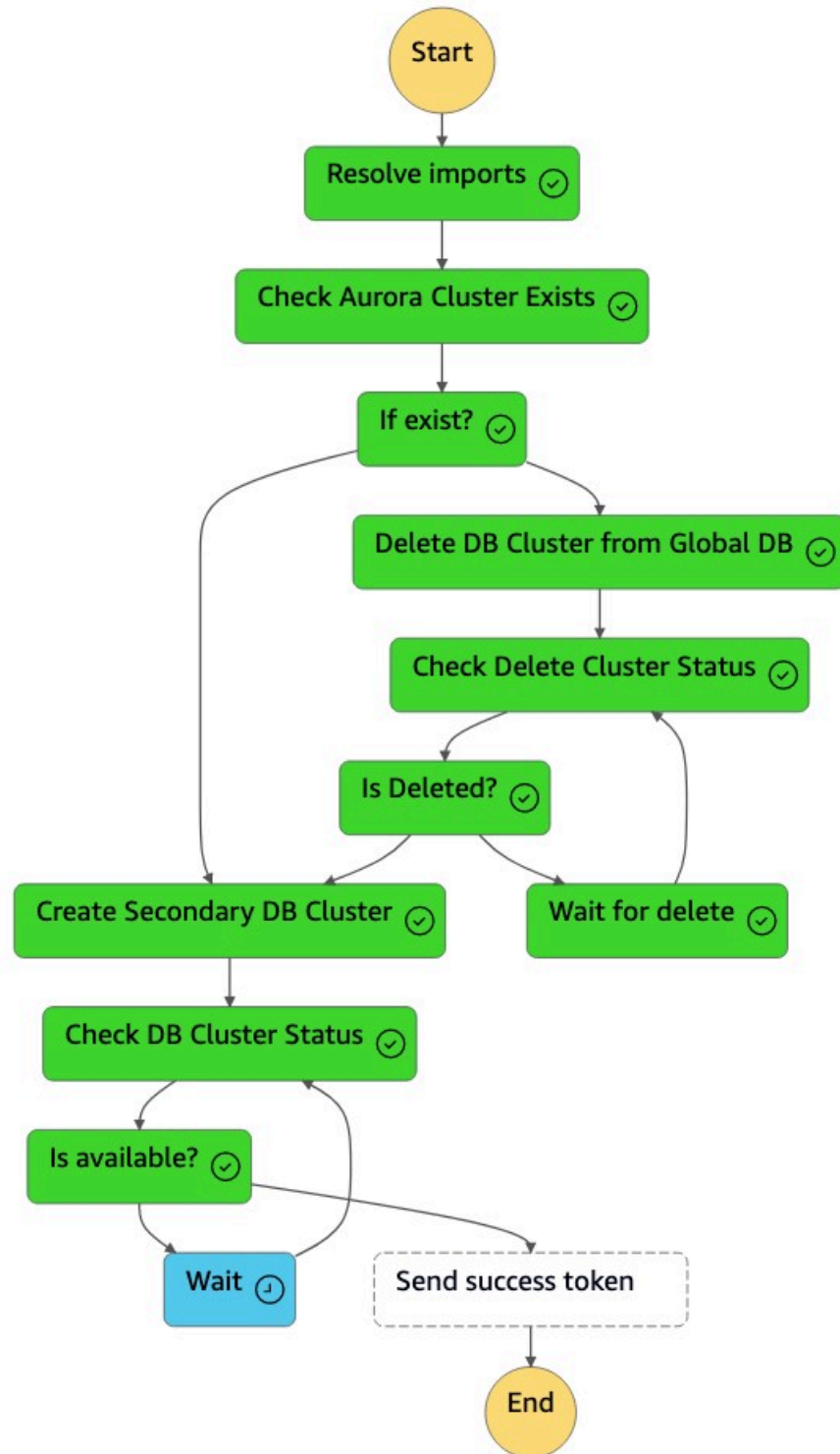
6. La máquina de DR Orchestrator FAILBACK estados lee el tipo de recurso como `CreateAuroraSecondaryDBCluster` y llama a la máquina de `dr-orchestrator-stepfunction-create-Aurora-Secondary-cluster` estados.



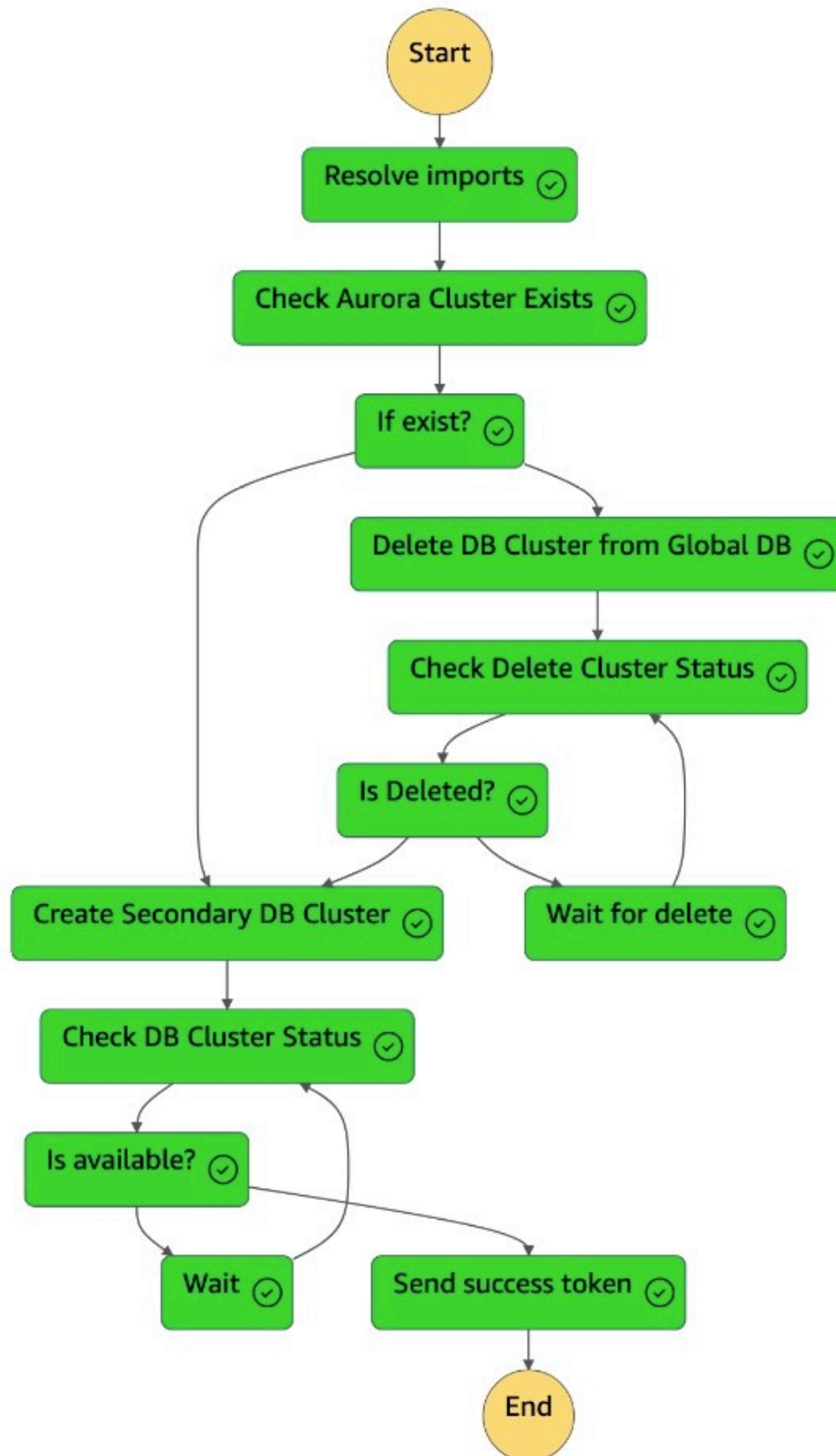
7. La máquina de `dr-orchestrator-stepfunction-create-Aurora-Secondary-cluster` estados elimina el clúster existente (`dbcluster-01`) de la antigua región principal (`us-east-1`).



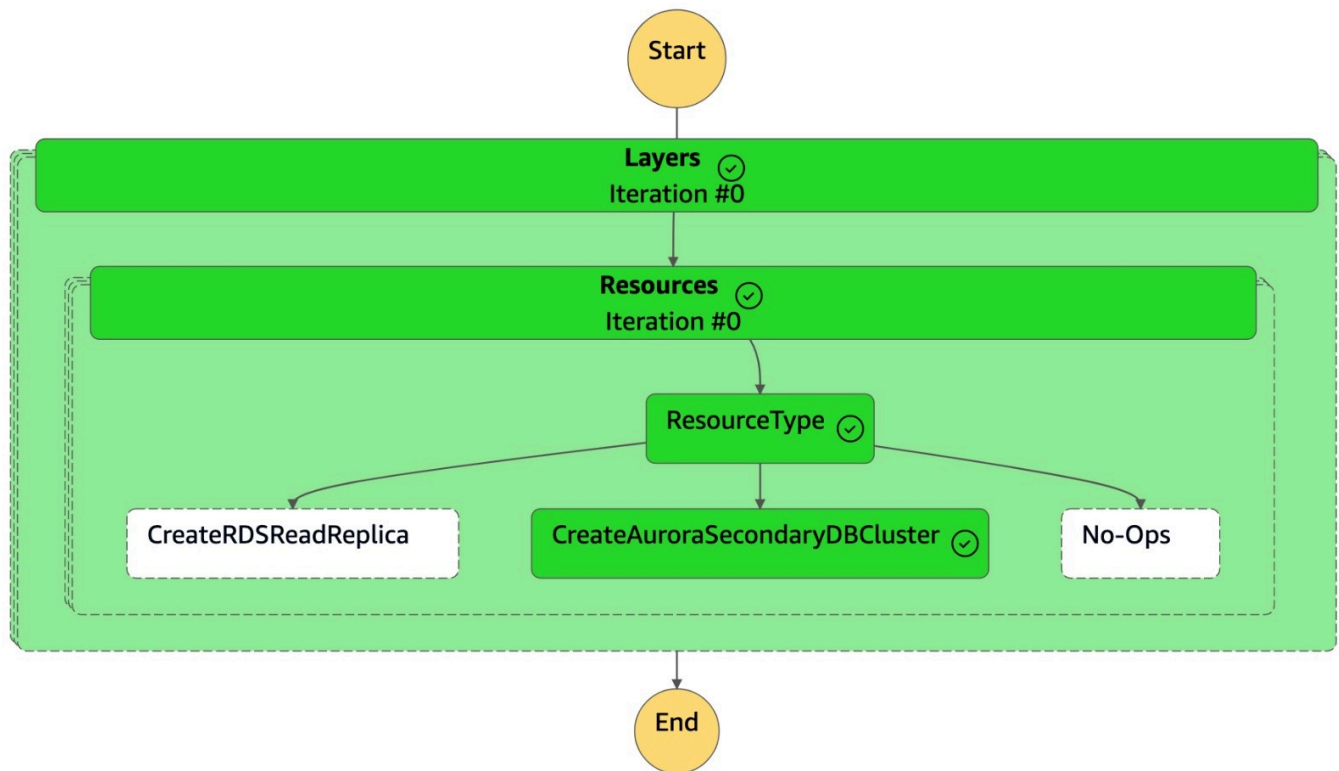
8. Tras eliminar el clúster (dbcluster-01), la máquina de estado crea un nuevo clúster (dbcluster-01) junto con la instancia de base de datos y se une a la base de datos global Aurora como clúster secundario para atender cargas de trabajo de solo lectura.



9. Cuando el clúster secundario está disponible, la máquina de `dr-orchestrator-stepfunction-create-Aurora-Secondary-cluster` estados se completa y envía un token de éxito a la máquina de DR `Orchestrator Failback` estados.



10 La máquina de `dr-orchestrator-stepfunction-FAILBACK` estados está completa.



11 Puede comprobar la base de datos global de Aurora en la consola de Amazon RDS.

[Si desea reubicar el clúster de base de datos principal en us-east-1, puede seguir los pasos que se mencionan en la sección **Switchover**.](#)

Implementación de la solución

Tras configurar los requisitos previos, siga las instrucciones del patrón [DR Orchestrator](#) Framework. Se incluyen instrucciones para Amazon RDS Aurora y Amazon ElastiCache. Para probar la solución, revise los requisitos previos y las limitaciones e implemente bases de datos de ejemplo en AWS.

Requisitos previos

- Estoy activo [Cuenta de AWS](#) para implementar esta solución.
- Permisos para crear buckets de [Amazon Simple Storage Service \(Amazon S3\)](#) y los recursos utilizados en las plantillas. [AWS CloudFormation](#)
- Antes de ejecutar la máquina de DR Orchestrator FAILBACK estados:
 - [Modifique manualmente la RDS instancia](#) o [modifique el clúster de base de datos Amazon Aurora](#) e inhabilite `DeletionProtection`.
 - Realice una [instantánea de la base](#) de datos antes de eliminar la RDS instancia de Amazon o el clúster de Aurora si tiene previsto realizar un análisis de la causa raíz (RCA).

Limitaciones

- La disponibilidad y el soporte de las funciones varían según las versiones específicas de cada motor de base de datos y entre ellas Regiones de AWS. Para obtener más información sobre la disponibilidad de funciones y regiones para la replicación entre regiones, consulte [Réplicas de lectura entre regiones](#).
- Esta solución se probó con una instancia de base de datos principal con una réplica de lectura. Si desea utilizar más de una réplica de lectura, pruebe la solución minuciosamente antes de implementarla en un entorno de producción.
- En el caso de AmazonRDS, la solución solo es compatible con los siguientes motores: [Amazon RDS for My SQL](#), [Amazon RDS for Postgre SQL](#) y [Amazon for RDS MariaDB](#).

Implemente bases de datos de ejemplo en AWS

Para demostrar la solución de recuperación ante desastres entre regiones para bases de datos AWS, puede implementar [Amazon RDS for My SQL](#), [Amazon Aurora My SQL -Compatible Edition](#) y [Amazon ElastiCache \(RedisOSS\)](#) - Global Datastore. [Siga las instrucciones del App-Stack README](#)

[archivo del repositorio -databases. aws-cross-region-dr](#) GitHub En este ejemplo, la región principal es us-east-1 y la región secundaria es us-west-2.

La pila de aplicaciones utiliza la sección de AWS CloudFormation [salidas](#) para exportar los parámetros que utiliza DR Orchestrator Framework para automatizar la conmutación por error y la conmutación por recuperación de las bases de datos. AWS

Preguntas frecuentes

Encuentre respuestas a las preguntas sobre DR Orchestrator Framework para la conmutación por error y la conmutación por recuperación.

¿Cuáles son el RPO y el RTO que puedo lograr con este enfoque?

Para obtener información sobre el objetivo de tiempo de recuperación (RTO) y el objetivo de punto de recuperación (RPO), consulte la estrategia Estrategia de [recuperación ante desastres para bases de datos sobre AWS](#)

¿Es obligatorio utilizar variables de AWS CloudFormation exportación?

No, puede pasar directamente el valor de la base de datos global de Amazon Aurora o de la instancia de base de datos de Amazon RDS directamente en formato JSON (por ejemplo: - "RDSInstanceIdentifier": "rds-mysql-instance").

¿Puedo usar el flujo de trabajo de conmutación por error de DR Orchestrator para realizar la conmutación por error de más de una base de datos? AWS

Sí, puede pasar más de un recurso al archivo de parámetros de entrada para realizar la conmutación por error de más de una AWS base de datos. El siguiente ejemplo de código muestra una conmutación por error de una réplica de lectura de Amazon RDS for MySQL y un almacén de datos global de ElastiCache Amazon (Redis OSS) en paralelo:

```
{
  "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"
        }
    }
]
}

```

¿Cómo puedo evitar el InvalidParameterCombination error al ejecutar la máquina de estados FAILBACK de DR Orchestrator para Amazon RDS?

El texto completo del error es:

```

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

```

Para evitar el error, [modifique la instancia de RDS](#) deshabilitándola DeletionProtection antes de ejecutar la máquina de DR Orchestrator FAILBACK estados.

Siguientes pasos

Puede utilizar DR Orchestrator Framework para añadir un proceso de aprobación antes de iniciar las actividades de conmutación por error o conmutación por recuperación. Por ejemplo, al iniciar la conmutación por error durante un evento de DR, puede añadir una máquina de estados encima DR Orchestrator Failover para enviar una [notificación del Amazon Simple Notification Service \(Amazon SNS\) por correo electrónico](#). En cuanto se conceda la aprobación, se iniciará la actividad de conmutación por error.

La conmutación por error de aplicaciones también se puede integrar con DR Orchestrator Framework para cambiar la entrada de DNS y apuntar las aplicaciones al nuevo punto final de la instancia o el clúster de la base de datos.

Puede utilizar el [AWS X-Ray](#) rastreo para obtener la duración de la conmutación por error para calcular el RTO. Puede crear un panel de monitoreo sobre X-Ray.

Recursos

- [Replicación Regiones de AWS mediante el uso de almacenes de datos globales](#) (documentación de Amazon ElastiCache (Redis OSS))
- [Uso de la conmutación por error en una base de datos global de Amazon Aurora](#) (documentación de Aurora)
- [Automatice la administración de puntos finales de Amazon Aurora Global Database para la conmutación por error planificada y no planificada](#) (entrada del blog)
- [Recuperación ante desastres: 3 escenarios de conmutación por error para su base de datos global de Amazon Aurora con Terraform \(parte 2\)](#) (entrada del blog)
- [Estrategia de recuperación ante desastres para bases de datos AWS](#) (estrategia de orientación AWS prescriptiva)
- [Automatice la conmutación por error y la conmutación por recuperación entre regiones mediante DR Orchestrator Framework](#) (patrón de orientación prescriptiva)AWS

Historial del documento

En la siguiente tabla, se describen cambios significativos de esta guía. Si quiere recibir notificaciones de futuras actualizaciones, puede suscribirse a las [notificaciones RSS](#).

Cambio	Descripción	Fecha
Publicación inicial	—	3 de mayo de 2024

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.