



Automatize sua solução de DR para bancos de dados relacionais em AWS

AWS Recomendações



AWS Recomendações: Automatize sua solução de DR para bancos de dados relacionais em AWS

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

As marcas comerciais e imagens de marcas da Amazon não podem ser usadas no contexto de nenhum produto ou serviço que não seja da Amazon, nem de qualquer maneira que possa gerar confusão entre os clientes ou que deprecie ou desprestige a Amazon. Todas as outras marcas comerciais que não pertencem à Amazon pertencem a seus respectivos proprietários, que podem ou não ser afiliados, patrocinados pela Amazon ou ter conexão com ela.

Table of Contents

Introdução	1
Objetivos	1
Visão geral do	3
O processo de failover	4
Arquitetura de FAILOVER do DR Orchestrator	4
O processo de failback	5
Arquitetura FAILBACK do DR Orchestrator	5
FAILOVER DO DR Orchestrator	7
Detalhes do parâmetro	8
FAILBACK do DR Orchestrator	9
Máquinas de estados do Aurora e Step Functions	12
Estado estacionário	12
Estado do evento	12
Transição	13
Failover manual não planejado	18
Failback	22
Implante a solução	31
Perguntas frequentes	33
Quais são o RPO e o RTO que posso alcançar usando essa abordagem?	33
É obrigatório usar variáveis de AWS CloudFormation exportação?	33
Posso usar o fluxo de trabalho de FAILOVER do DR Orchestrator para fazer failover em mais de um banco de dados? AWS	33
Como posso evitar o InvalidParameterCombination erro ao executar a máquina de estado FAILBACK do DR Orchestrator para o Amazon RDS?	34
Próximas etapas	35
Recursos	36
Histórico do documentos	37
.....	xxxviii

Automatize sua solução de DR para bancos de dados relacionais em AWS

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

Maio de 2024 ([histórico do documento](#))

Ao planejar uma [estratégia de recuperação de desastres \(DR\)](#) na Amazon Web Services (AWS), você pode implementar uma solução de DR para os bancos de dados em sua organização. Você pode automatizar a solução de DR configurando uma arquitetura orientada por eventos para realizar o failover das instâncias do banco de dados. Os benefícios dessa abordagem incluem o seguinte:

- Reduzindo os erros humanos causados pela coordenação manual
- Fornecendo o objetivo de tempo de recuperação (RTO) mais curto possível em alguns casos
- Facilitando testes repetíveis de sua solução de DR

Sem automação, uma interrupção em seu sistema primário Região da AWS exigiria recursos em sua organização para executar manualmente as etapas em um caderno de execução. Essas etapas podem incluir vários processos, como promover réplicas de leitura do Amazon Relational Database Service (Amazon RDS) no Região da AWS secundário para servir como a nova instância primária do banco de dados. AWS fornece o [AWS SDK for Python \(Boto3\) para APIs automatizar essas ações, que podem formar os alicerces de uma solução de DR](#) para as necessidades da sua organização.

Este guia discute um orquestrador de DR centralizado e orientado por manifestos que pode ajudá-lo a obter uma solução de DR para AWS bancos de dados dentro do mesmo. Regiões da AWS Conta da AWS A decisão de realizar um failover ainda precisaria de intervenção humana e está fora do escopo da automação.

O público-alvo deste guia são arquitetos de aplicativos, arquitetos de infraestrutura, arquitetos de banco de dados e consultores de banco de dados.

Objetivos

A implementação da solução DR Orchestrator Framework pode ajudá-lo a alcançar os seguintes resultados:

- Reduza os erros manuais durante uma interrupção — o desempenho manual dos processos de DR pode estar sujeito a erros. Automatizar as ações durante as fases de uma atividade de DR reduz os erros manuais e ajuda a garantir um failover ou failback mais suave.
- Gerencie DR em grande escala — Se sua organização tem uma grande frota de bancos de dados, a automação predial para DR ajuda você a escalar sua solução de DR. Você pode automatizar sua solução de DR AWS de banco de dados usando o DR Orchestrator Framework. O DR Orchestrator automatiza as etapas de failover ou failback de AWS bancos de dados usando um arquivo de parâmetros no formato JSON.

Visão geral do DR Orchestrator Framework

O DR Orchestrator Framework fornece uma solução de um clique para orquestrar e automatizar a recuperação de desastres entre regiões para bancos de dados. AWS Ele usa [AWS Step Functions](#) e [AWS Lambda](#) executa as etapas necessárias durante o failover e o failback. As máquinas de estado do Step Functions fornecem a base para a tomada de decisões dentro do design do orquestrador. As operações de API para realizar ações de failover ou failback são codificadas em funções Lambda que são chamadas de dentro da máquina de estado. As funções do Lambda executam [AWS SDK para Python \(Boto3\)](#) APIs para interagir com bancos de dados. AWS

O DR Orchestrator Framework contém duas máquinas de estado principais que correspondem às fases de failover e failback.

Para o Amazon RDS, a fase de failover promove uma réplica de leitura do RDS entre regiões em uma instância de banco de dados independente. Para o Amazon Aurora, quando a região principal fica inativa durante uma interrupção rara e inesperada, seu nó de gravação não está disponível. A replicação entre o nó do gravador e os clusters secundários é interrompida. Você deve separar o cluster secundário do banco de dados global e promovê-lo como um cluster autônomo. Os aplicativos podem se conectar e enviar tráfego de gravação para o cluster autônomo. Você pode usar esse mesmo processo para [alternar o cluster de](#) banco de dados primário do banco de dados global para as regiões secundárias. Use essa abordagem para cenários controlados, como os seguintes:

- Manutenção operacional
- Procedimentos operacionais planejados
- Promoção de um cluster secundário Amazon ElastiCache (Redis OSS) como seu novo cluster primário

A fase de failback estabelece a replicação ativa dos dados entre uma região primária ativa e uma nova região secundária.

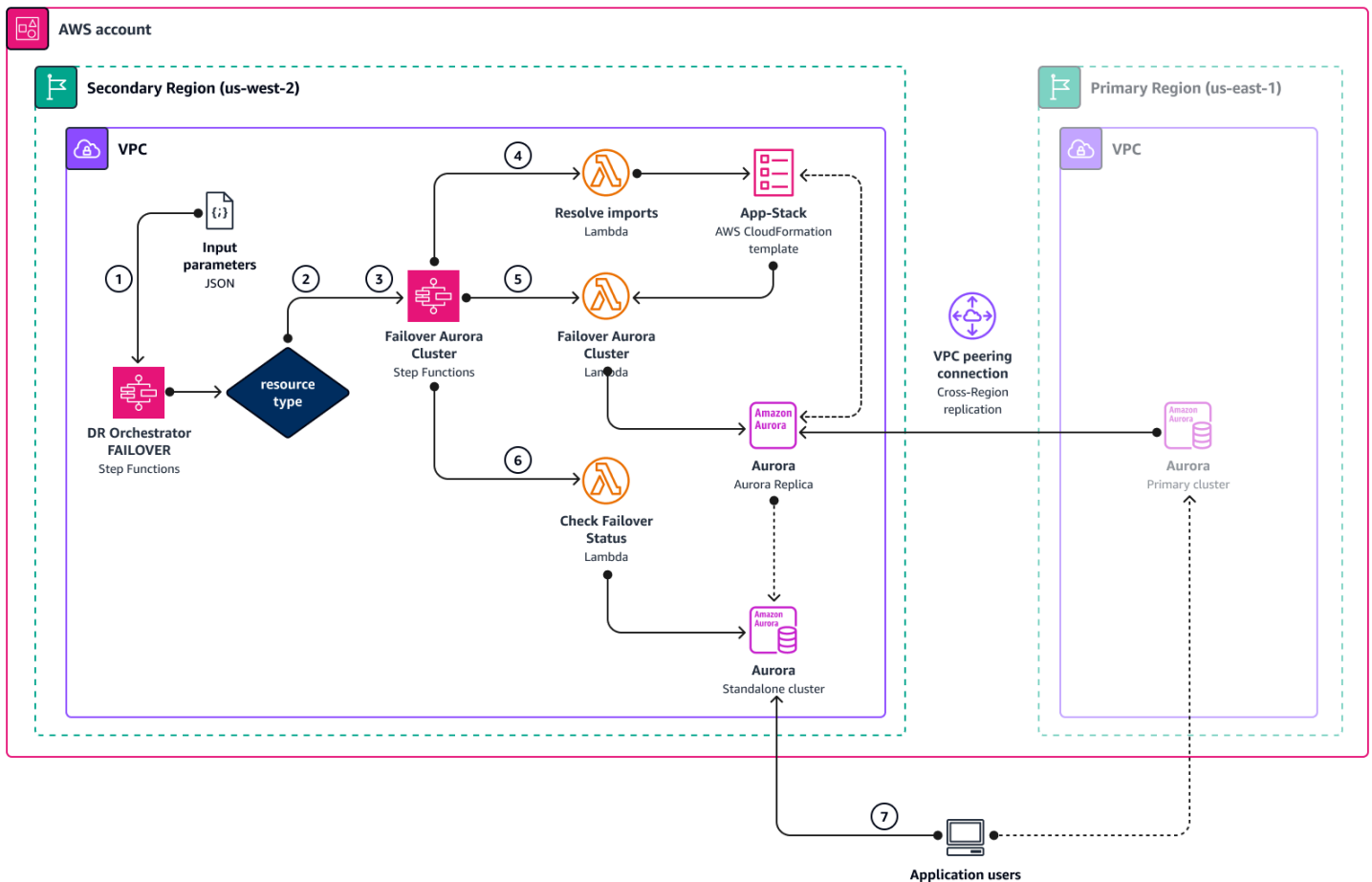
É fundamental entender que o DR Orchestrator se aplica somente aos bancos de dados. Todos os aplicativos que fazem referência a esses bancos de dados e estão na mesma região podem precisar de uma solução de failover separada e em tandem. Após o failover dos bancos de dados para a região secundária, os aplicativos precisam ser atualizados para se conectarem às novas instâncias do banco de dados, que servirão como fonte de dados.

O processo de failover

Para realizar um failover, execute a máquina de DR **Orchestrator FAILOVER** estado. Nesse estágio, um banco de dados secundário já está presente na região secundária, seja como uma réplica de leitura (Amazon RDS) ou como um cluster secundário (Amazon Aurora). Quando você executa a máquina de DR **Orchestrator FAILOVER** estado, ela promove que o banco de dados secundário se torne o principal.

Arquitetura de FAILOVER do DR **Orchestrator**

O diagrama a seguir mostra os conceitos do processo de failover do Amazon Aurora ao usar o DR Orchestrator. O Amazon Aurora e a Amazon ElastiCache usam o mesmo fluxo de trabalho, mas com máquinas de estado e funções Lambda diferentes.



1. A máquina de DR **Orchestrator FAILOVER** estado lê os parâmetros JSON de entrada.

2. Com base no `resourceType` parâmetro, a máquina de estado chama outras máquinas de estado: `Promote RDS Read Replica`, `Failover Aurora Cluster`, ou `Failover ElastiCache`. Se mais de um recurso for passado na entrada, essas máquinas de estado serão executadas paralelamente.
3. A máquina de `Failover Aurora Cluster` estado chama as funções Lambda em cada uma das três etapas a seguir.
4. A função `Resolve imports Lambda` é resolvida "`! import <export-variable-name>`" com os valores reais do modelo. App-Stack AWS CloudFormation
5. A função **`Failover Aurora Cluster`** Lambda promove a réplica de leitura como uma instância de banco de dados independente.
6. A função **`Check Failover Status`** Lambda verifica o status da instância de banco de dados promovida. Depois que o status estiver DISPONÍVEL, a função Lambda envia um token de sucesso de volta para a máquina de estado de chamada e é concluída.
7. Você pode redirecionar seus aplicativos para o banco de dados autônomo na Região DR (`us-west-2`), que agora é o banco de dados principal.

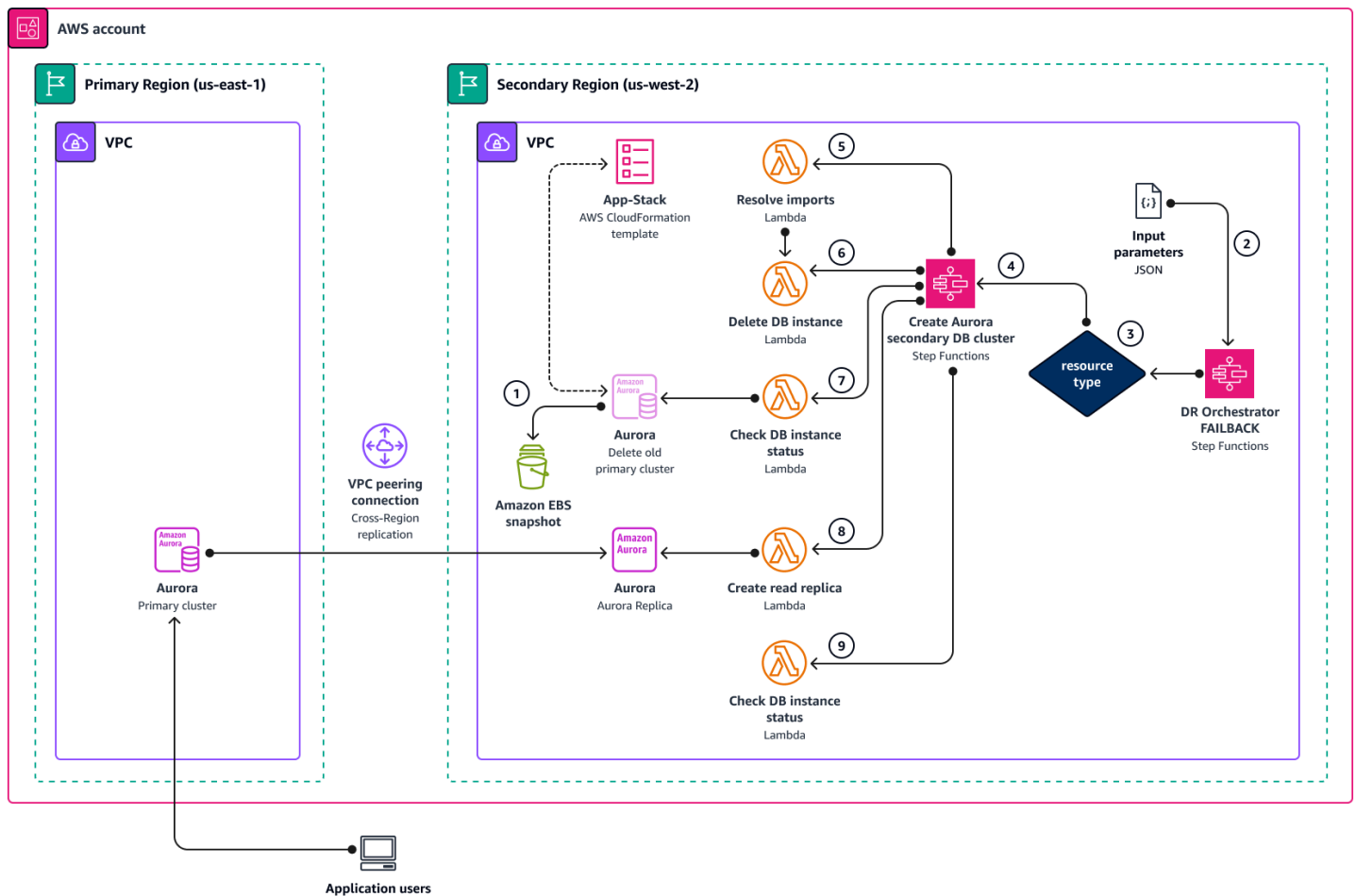
O processo de failback

Depois que sua antiga região primária (`us-east-1`) estiver ativa novamente, você poderá voltar para ela, para que o banco de dados `us-east-1` se torne o principal novamente. Para iniciar o failback, execute a máquina de DR `Orchestrator FAILBACK` estado. Como o nome indica, essa máquina de estado começa a replicar as alterações em sua nova região primária (`us-west-2`) de volta para a antiga região primária (`us-east-1`), que atua como a secundária atual.

Depois que a replicação for estabelecida entre as duas regiões, você poderá iniciar o failback. Para retornar e retornar à sua região primária original (`us-east-1`), execute a máquina de DR `Orchestrator FAILOVER` estado na região secundária atual (`us-east-1`) para promovê-la à região primária.

Arquitetura FAILBACK do **DR Orchestrator**

O diagrama a seguir mostra os conceitos do processo de failback para o Amazon Aurora ao usar o DR Orchestrator.



1. Antes de iniciar o failback, faça um DB snapshot manual para usar ao realizar a análise de causa raiz (RCA).

Além disso, desative o DeletionProtection para o cluster Aurora na região primária anterior (us-east-1).

2. A máquina de DR Orchestrator FAILBACK estado lê os parâmetros JSON de entrada.

3. Com base no resourceType, a máquina de DR Orchestrator FAILBACK estado chama a máquina de Create Aurora Secondary DB Cluster estado.

4. A máquina de Create Aurora Secondary DB Cluster estado chama as funções Lambda em cada uma das cinco etapas a seguir.

5. A função Resolve import Lambda é resolvida `! import <export-variable-name>` com os valores reais do modelo. App-Stack CloudFormation

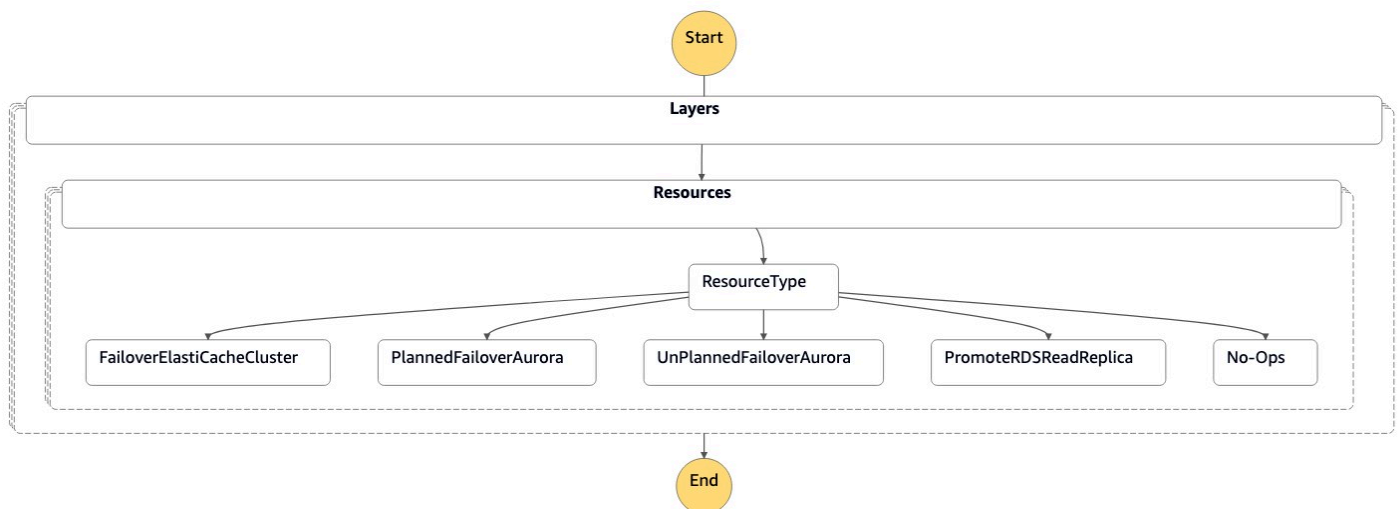
6. A função Delete DB Instance Lambda exclui a instância primária anterior.

7. A função Check DB instance status Lambda verifica Delete DB Instance status até que o banco de dados seja excluído.
8. A função Create Read Replica Lambda cria uma réplica de leitura na região secundária a partir da instância de banco de dados que está na nova região primária.
9. A função Check DB instance status Lambda verifica o status da instância de banco de dados da réplica de leitura. Quando o status é DISPONÍVEL, a função Lambda envia um token de sucesso de volta para a máquina de estado de chamada, que é concluída.

FAILOVER DO DR Orchestrator

Use a máquina de DR Orchestrator FAILOVER estado no evento de DR quando a região primária (us-east-1) estiver inativa ou durante eventos planejados, como manutenção operacional.

A função pode ser chamada para realizar o failover de um ou vários bancos de dados em paralelo.



A máquina de estado aceita parâmetros no formato JSON, conforme mostrado no código a seguir:

```

{
  "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"
        }
    }
]
```

Detalhes do parâmetro

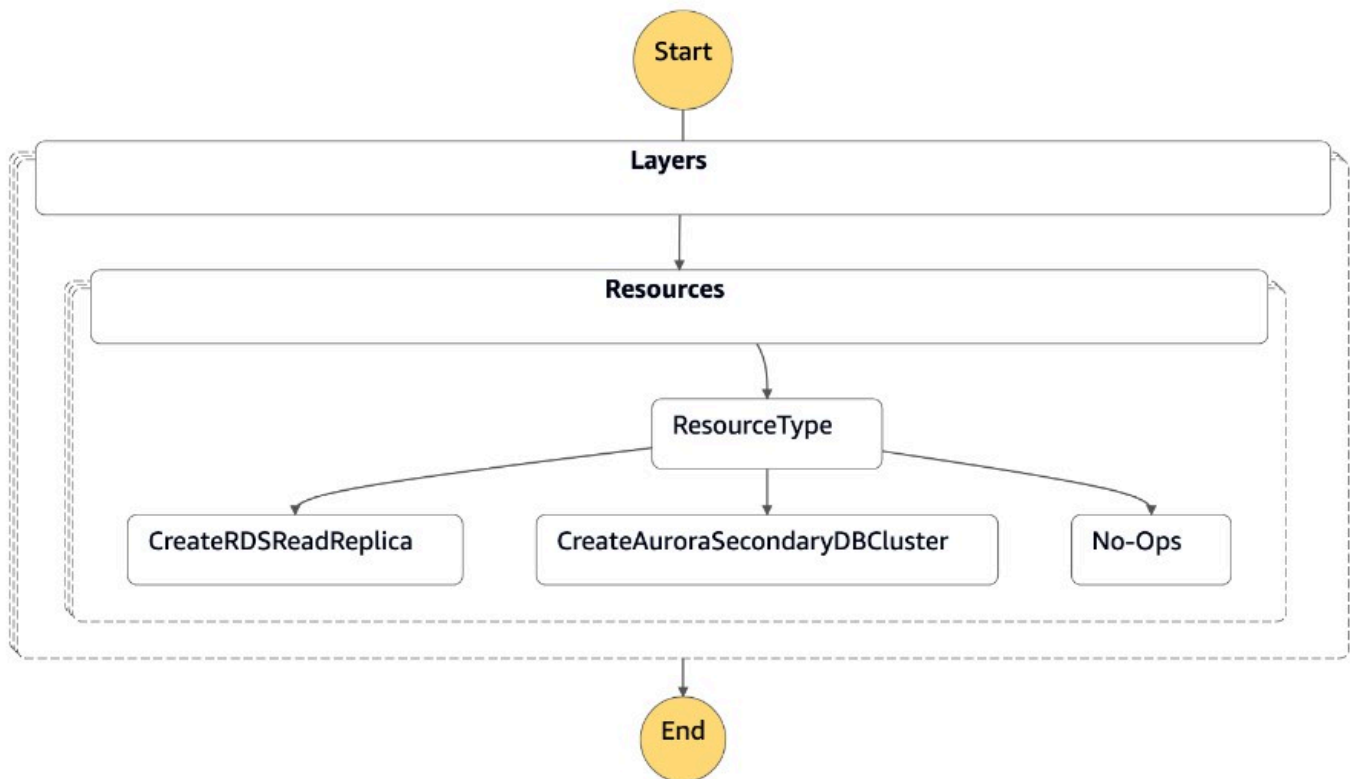
A tabela a seguir mostra os parâmetros usados pela máquina de DR Orchestrator FAILOVER estado.

Nome do parâmetro	Description	Valores esperados
layer(obrigatório: número)	A sequência de processamento. Todos os recursos definidos na camada 1 devem ser executados antes que os recursos da camada 2 sejam executados.	1 ou 2, e assim por diante
recursos (obrigatório: matriz de dicionário)	Todos os recursos em uma única camada são executados paralelamente.	<pre>{ "resourceType": " String", "resourceName": "St ring", "parameters": {</pre>

Nome do parâmetro	Description	Valores esperados
		<pre>"<param1>":"<!Import cft-output-1">, }</pre>
resourceType (obrigatório: string)	Tipo do recurso para identificar o recurso	PromoteRDSReadReplica ou FailoverElasticCacheCluster
resourceName (opcional: string)	Para identificar a qual portfólio de aplicativos esses recursos pertencem	Promote RDS for MySQL Read Replica
parâmetros (obrigatório: matriz de dicionário)	Lista de parâmetros necessários para fazer failover ou failback do AWS banco de dados	<pre>{ "<param1>": "<!Import cft-output-1">, "<param2>": "<!Import cft-output-2">, }</pre>

FAILBACK do DR Orchestrator

Use a máquina de DR Orchestrator FAILBACK estado após o evento de DR, quando a antiga região primária (us-east-1) estiver ativa. Você pode criar a [réplica de leitura](#) para o Amazon RDS na antiga região primária a partir da nova região primária (us-west-2) para estar em conformidade com sua estratégia de DR. Como esse é um evento planejado, você pode agendar essa atividade no fim de semana ou fora do horário comercial de pico com um tempo de inatividade estimado.



A máquina de estado aceita parâmetros no formato JSON, conforme mostrado no código a seguir:

```

{
  "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 do Aurora e Step Functions

Esta seção aborda as máquinas de processo e estado específicas para o failover e o failback dos clusters do Amazon Aurora. Os clusters são configurados como um banco de dados global.

Note

Para fins de demonstração, este exemplo usa a Edição Aurora MySQL-Compatible . Você pode usar etapas semelhantes para a Edição Aurora PostgreSQL-Compatible .

Estado estacionário

No estado estável, um banco de dados global Amazon MySQL-Compatible Aurora `dr-globaldb-cluster-mysql ()` foi criado com dois clusters de banco de dados. O primeiro cluster de banco de dados (`db-cluster-01`) foi criado no primary Região da AWS (`us-east-1`) para atender à read/write carga de trabalho. O segundo cluster de banco de dados (`db-cluster-02`) foi criado na região secundária (`us-west-2`) para atender à carga de trabalho somente para leitura.

Além de fornecer a solução de DR, você pode reduzir a carga em seu cluster de banco de dados primário roteando consultas de leitura de seus aplicativos para o cluster de banco de dados secundário. Cada um desses clusters contém uma instância de banco de dados chamada `dbcluster-01-use1-instance-1` e `dbcluster-02-usw2-instance-2`, respectivamente.

Estado do evento

Ao usar um banco de dados global do Amazon Aurora, você pode planejar e se recuperar de um desastre com bastante rapidez. A recuperação de desastres geralmente é medida usando valores para objetivo de tempo de recuperação (RTO) e objetivo de ponto de recuperação (RPO). Para obter mais informações, consulte [Uso de alternância ou failover em um banco de dados global do Amazon Aurora](#).

Com um banco de dados global Aurora, há duas abordagens diferentes para o failover:

- Transição (failover planejado gerenciado)
- Failover (failover manual não planejado ou desanexar e promover)

Transição

A transição é destinada a ambientes controlados, como manutenção operacional e outros procedimentos operacionais planejados. Ao usar um failover planejado gerenciado, você pode realocar o cluster de banco de dados principal do seu banco de dados global Aurora para uma das regiões secundárias. Como o switchover espera até que os clusters de banco de dados secundários sejam sincronizados com o banco de dados primário, o RPO é 0 (sem perda de dados). Para saber mais, consulte [Execução de transições para bancos de dados globais do Amazon Aurora](#).

A máquina de `dr-orchestrator-stepfunction-FAILOVER` estado é invocada durante o estado do evento para alternar seu cluster primário para a região secundária escolhida (`us-west-2`).

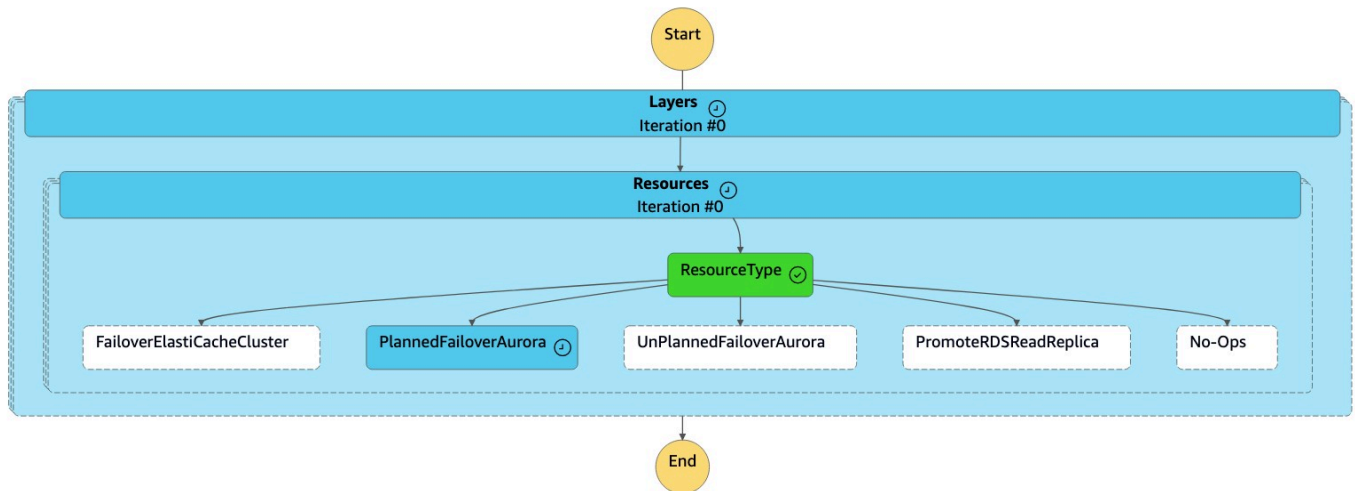
Para realizar a transição, faça o seguinte:

1. Faça login no Console de gerenciamento da AWS.
2. Altere a região para a região DR (`us-west-2`).
3. Navegue até Services e escolha Step Functions.
4. Navegue até a máquina de `dr-orchestrator-stepfunction-FAILOVER` estado.
5. Escolha Iniciar execução e insira o seguinte código JSON na Input - optional seção:

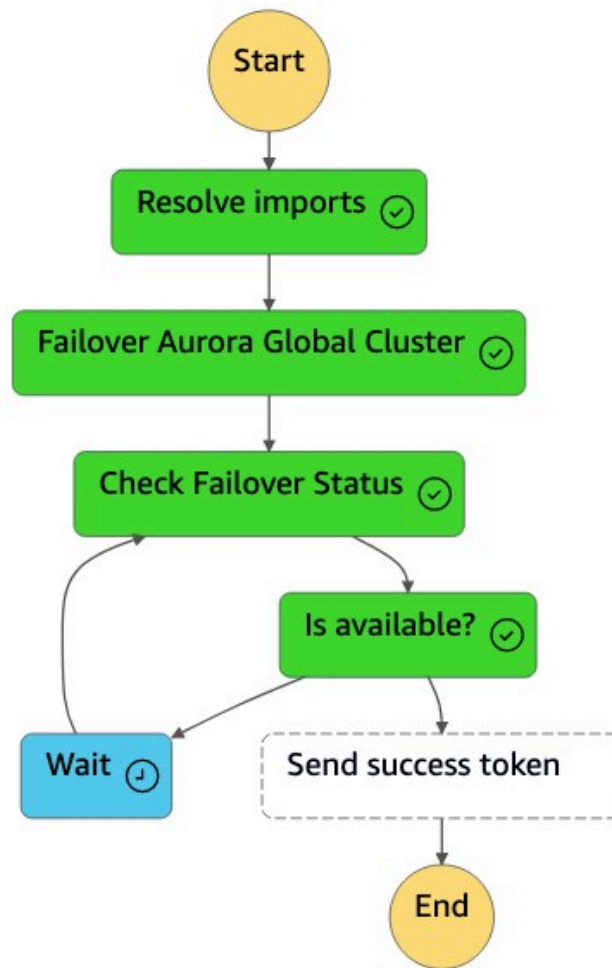
```
{
  "StatePayload": [
    {
      "layer": 1,
      "resources": [
        {
          "resourceType": "PlannedFailoverAurora",
          "resourceName": "Switchover (planned failover) of Amazon Aurora global
databases (MySQL)",
          "parameters": {
            "GlobalClusterIdentifier": "!Import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

```
}
```

6. A máquina de `dr-orchestrator-stepfunction-FAILOVER` estado lê o tipo de recurso como `PlannedFailoverAuroraMySQL` e chama a máquina de `dr-orchestrator-stepfunction-planned-Aurora-failover` estado para fazer o failover do banco de dados global Aurora.



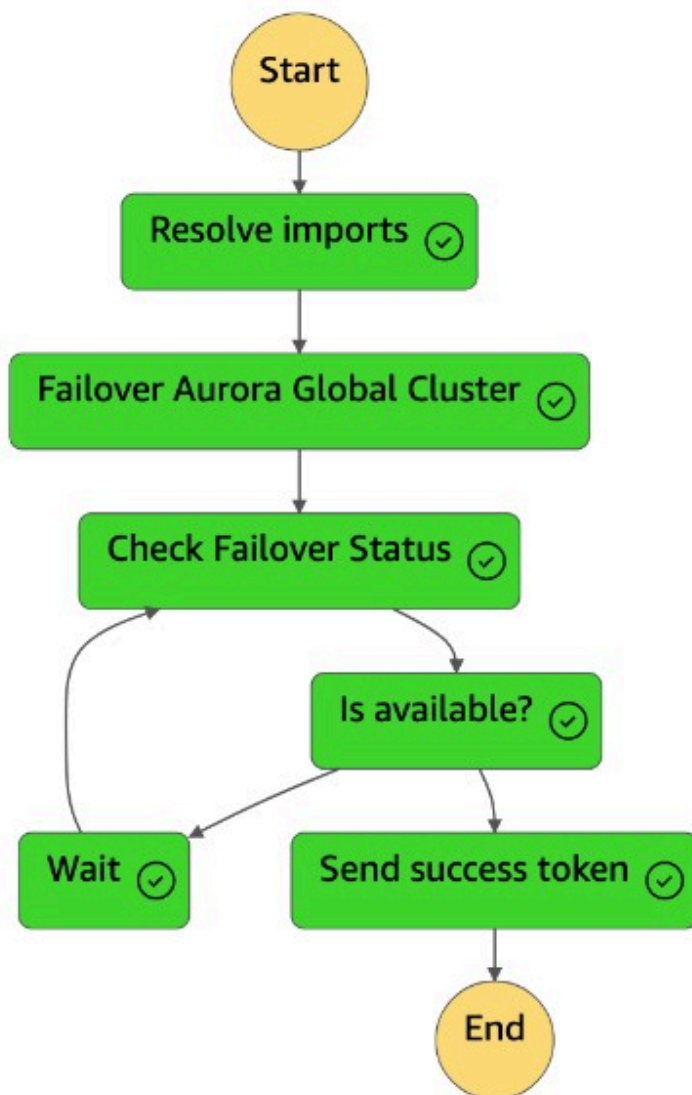
7. A máquina de `dr-orchestrator-stepfunction-planned-Aurora-failover` estado executa as seguintes etapas para alternar a função do banco de dados MySQL-Compatible global Aurora.



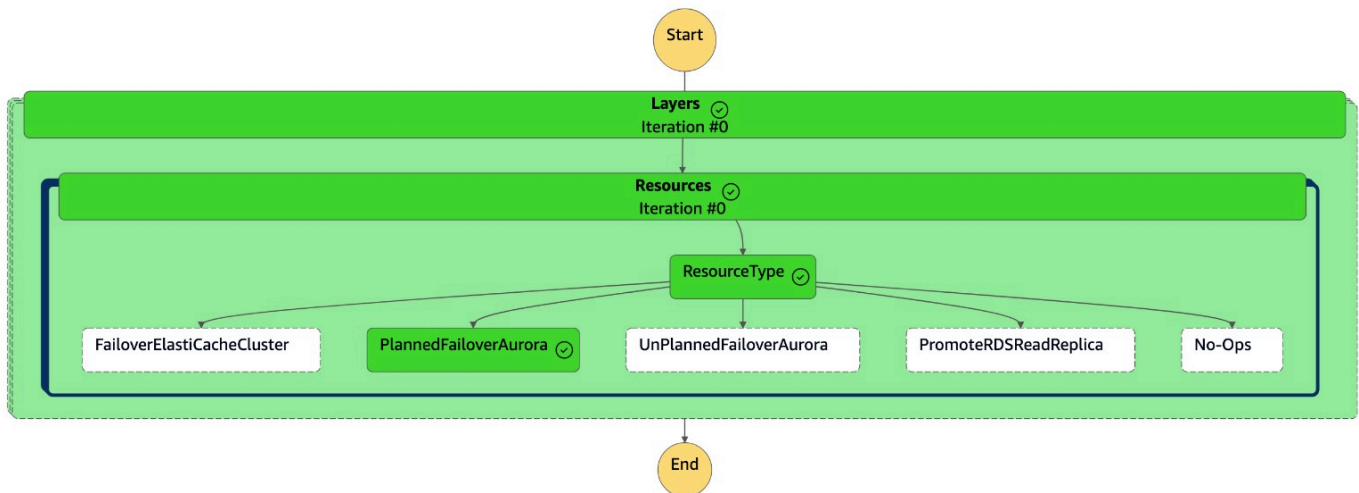
Etapa	Description	Valores esperados
Resolver importações	Uma função Lambda substitui <code>!Import <variable name></code> valores pelo nome real.	<code>"!Import dr-global db-cluster-mysql-global-identifier"</code> é substituído por <code>"dr-global db-cluster-mysql"</code> .

Etapa	Description	Valores esperados
Faça o failover do Aurora Global Cluster	Uma função Lambda chama as APIs failover_global_cluster Boto3 para fazer o failover do banco de dados global Aurora.	<pre>{ 'GlobalCluster': { 'GlobalClusterIdentifier': 'dr-global-cluster-mysql', 'GlobalClusterResourceId': 'cluster-cce7f9bec2846db4', 'GlobalClusterArn': 'arn:aws:rds::xxx', 'Status': 'failing-over', } }</pre>
Verifique o status do failover	Uma função Lambda chama as APIs Boto3 describe_db_clusters para verificar o status do failover.	modificando, disponível
Envie tokens de sucesso	Uma função Lambda chama as APIs de Boto3 send_task_success e envia um token de sucesso de volta para a máquina de estado. DR Orchestrator Failover	h7XRiCdLtd/83p1E0dMccoxIzFhglsdkzpK9mBVKZsp7d9yrT1W

- Navegue até o console do Amazon RDS. Em Status, os valores do banco de dados global do Aurora mudarão de Disponível para Alternância ou Modificação.
- Depois que a máquina de `dr-orchestrator-stepfunction-planned-Aurora-failover` estado é concluída, ela envia um token de sucesso de volta para a máquina de `dr-orchestrator-stepfunction-FAILOVER` estado.



10A máquina de `dr-orchestrator-stepfunction-FAILOVER` estado está concluída.



No console, a função do cluster secundário (**dbcluster-02**) agora é cluster primário, e o cluster está pronto para atender às read/write cargas de trabalho. A função do cluster primário original (dbcluster-01) agora está listada como cluster secundário.

Failover manual não planejado

Em raras ocasiões, seu banco de dados global Aurora pode sofrer uma interrupção inesperada em seu banco de dados primário. Região da AWS Se isso acontecer, seu cluster de banco de dados Aurora primário e seu nó de gravador não estarão disponíveis, e a replicação entre o cluster primário e os secundários cessará. Para minimizar o tempo de inatividade (RTO) e a perda de dados (RPO), trabalhe rapidamente para realizar um failover entre regiões e reconstruir seu banco de dados global Aurora. Para obter mais informações, consulte [Recuperação de um banco de dados global do Amazon Aurora após uma interrupção não planejada](#).

A execução de um failover não planejado exige que você separe seu cluster secundário do banco de dados global Aurora. Antes de realizar o failover não planejado, interrompa as gravações do aplicativo em seu cluster de banco de dados Aurora primário. Depois que o failover for concluído com êxito, reconfigure o aplicativo para gravar no novo cluster de banco de dados primário. Essa abordagem ajuda a evitar a perda de dados. Também ajuda a evitar inconsistências de dados se o nó do gravador primário voltar a ficar on-line durante o processo de failover.

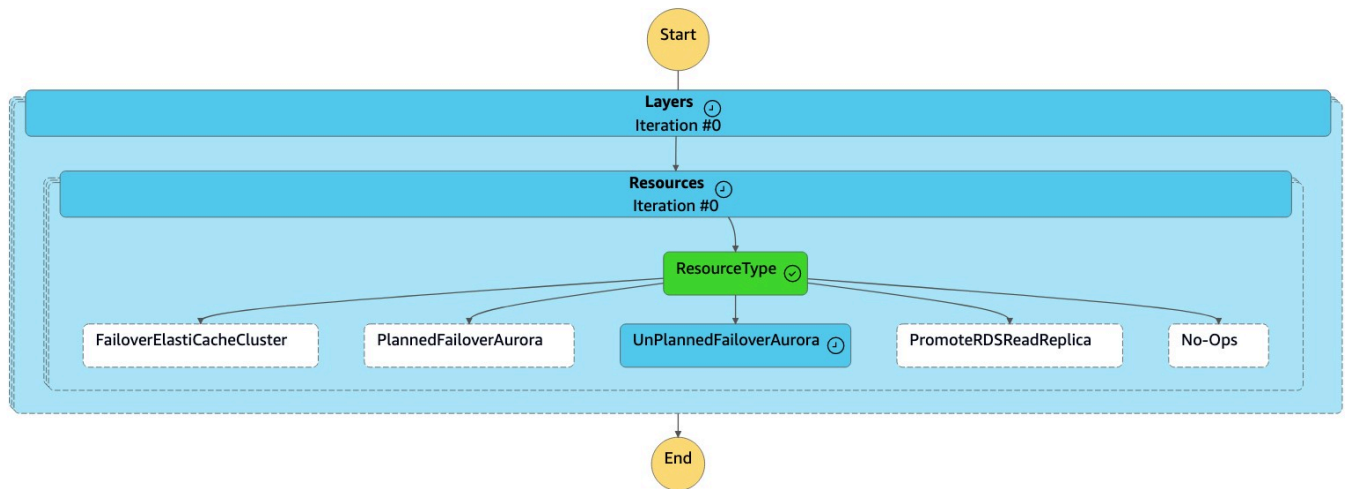
Para realizar o failover não planejado, ligue para a máquina de `dr-orchestrator-stepfunction-FAILOVER` estado. Neste exemplo, o cluster secundário (db-cluster-02) está na região DR (us-west-2) em estado estável.

Para realizar o failover, faça o seguinte:

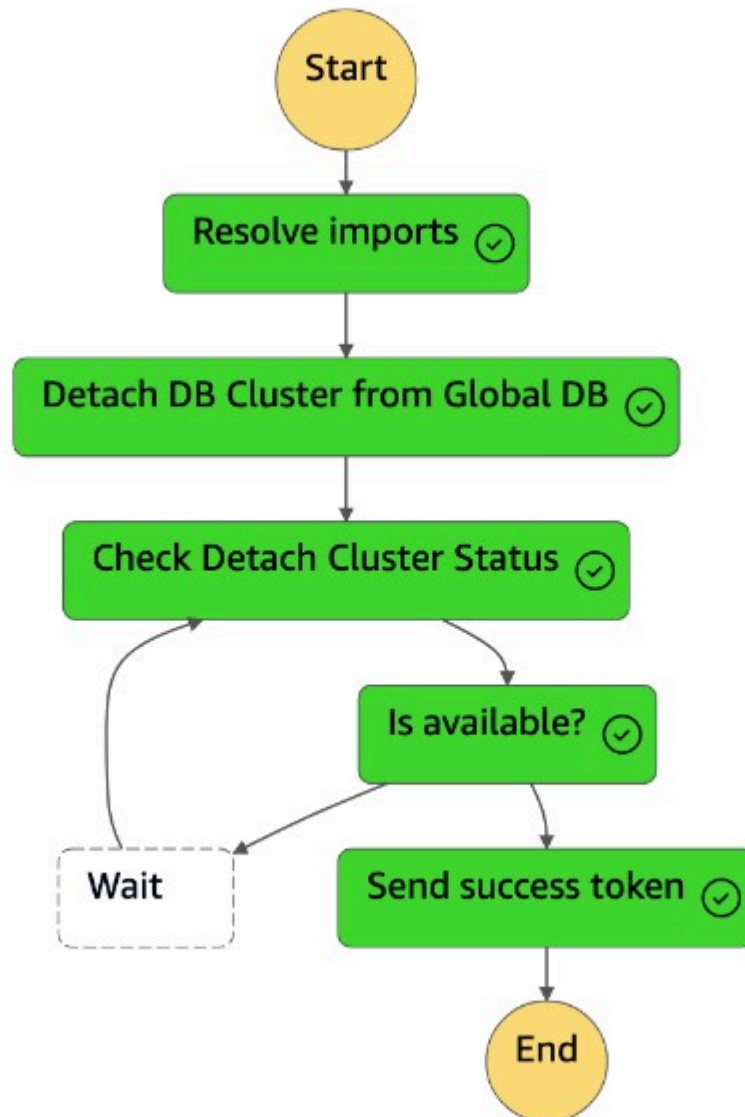
1. Faça login no console do .
2. Altere a região para a região DR (us-west-2).
3. Navegue até Services e escolha Step Functions.
4. Navegue até a máquina de dr-orchestrator-stepfunction-FAILOVER estado.
5. Escolha Iniciar execução e insira o seguinte código JSON na *Input - optional* seção, usando *UnPlannedFailoverAurora* 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"
          }
        }
      ]
    }
  ]
}
```

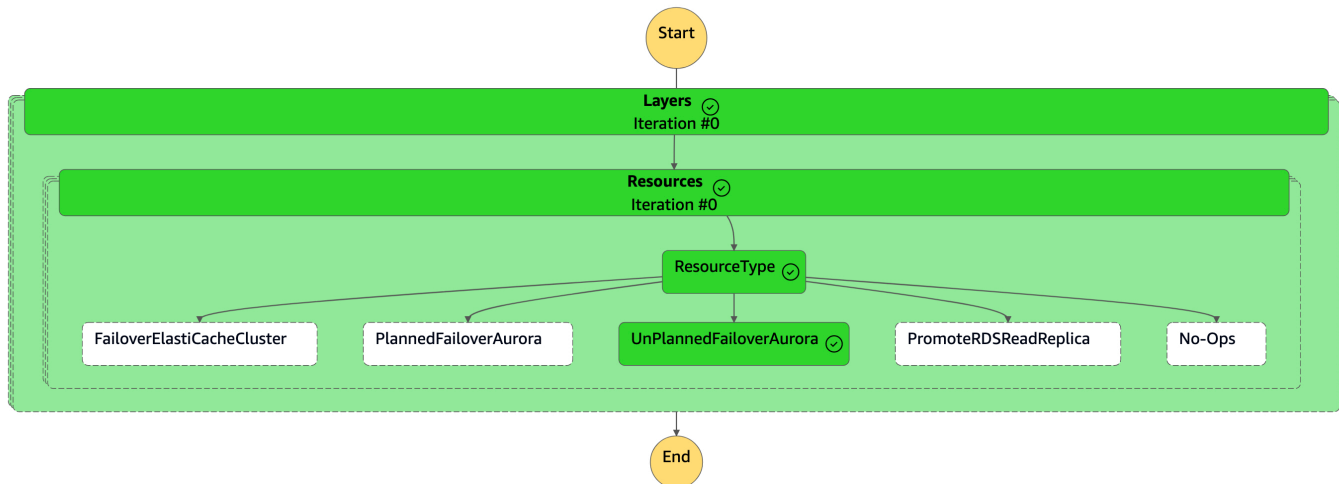
6. A máquina de dr-orchestrator-stepfunction-FAILOVER estado lê o tipo de recurso como UnPlannedFailoverAuroraMySQL e chama a tarefa Detach Cluster from Global Database da máquina de dr-orchestrator-stepfunction-unplanned-Aurora-failover estado.



7. A `Detach Cluster from Global Database` tarefa separa (remove) o cluster secundário do banco de dados global.



8. O cluster secundário (`dbcluster-02`) é promovido para se tornar um cluster autônomo e pode atender read/write cargas de trabalho.
9. A máquina de `dr-orchestrator-stepfunction-FAILOVER` estado está concluída.



10.O cluster secundário (dbcluster-02) é separado do banco de dados global Aurora e se torna um cluster independente para atender à carga de trabalho. read/writer

11.Reconfigure seu aplicativo para enviar todas as operações de gravação para esse novo cluster de banco de dados Aurora independente usando seu novo endpoint de cluster.

Failback

Um failback retorna seu banco de dados para o local principal original (ou novo) após a resolução de um desastre (ou evento programado). Quando a interrupção não planejada for resolvida, talvez você queira adicionar sua antiga região primária de volta ao banco de dados global da Aurora. Primeiro, você deve excluir o cluster de banco de dados existente da antiga região primária, criar um novo cluster de banco de dados da nova região primária e, em seguida, usar o processo de failover planejado gerenciado para mudar a função do novo cluster.

Isso pode ser considerado uma atividade planejada que você pode realizar fora do horário de pico ou em um fim de semana.

Você deve [modificar manualmente o cluster de banco de dados Amazon Aurora](#) e desativá-lo DeletionProtection antes de executar a máquina de DR Orchestrator FAILBACK estado da antiga região primária (us-east-1) porque ela foi criada com. DeletionProtection

O DR Orchestrator Framework usa a máquina de dr-orchestrator-stepfunction-FAILBACK estado para automatizar as etapas para excluir o cluster existente e criar um novo cluster na antiga região primária.

Para desativar `DeletionProtection`, faça o seguinte:

1. Faça login no console do .
2. Altere a região para a antiga região primária (us-east-1).
3. Navegue até o console do Amazon RDS, selecione o nome do cluster (dbcluster-01) e escolha Modificar.
4. Em Proteção contra exclusão, desmarque a caixa de seleção Ativar proteção contra exclusão e escolha Continuar.
5. Escolha Aplicar imediatamente e, em seguida, escolha Modificar cluster.

A máquina de DR Orchestrator FAILBACK estado é invocada durante o processo de failback da antiga região primária (us-east-1).

Para realizar o failback, faça o seguinte:

1. Faça login no console do .
2. Altere a região para a antiga região primária (us-east-1).
3. Navegue até Services e escolha Step Functions.
4. Navegue até a máquina de DR Orchestrator FAILBACK estado.
5. Escolha Iniciar execução e insira o seguinte código JSON na Input - optional seção:

```
{
  "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",

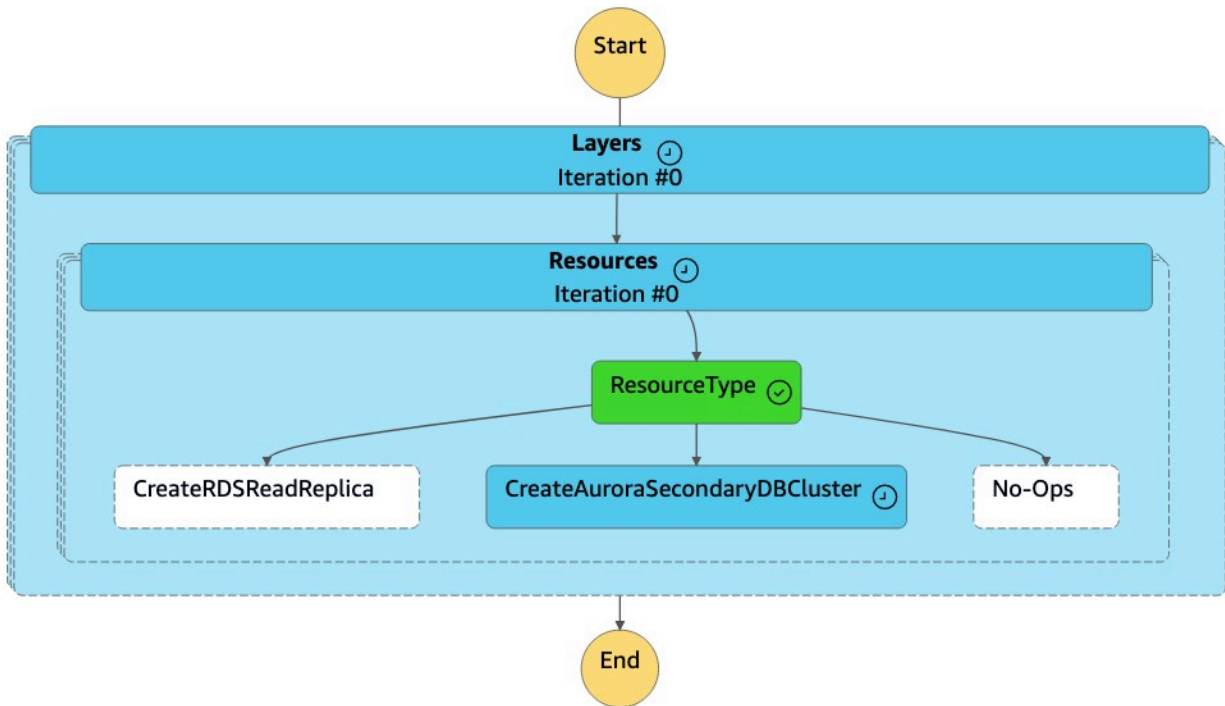
```

```

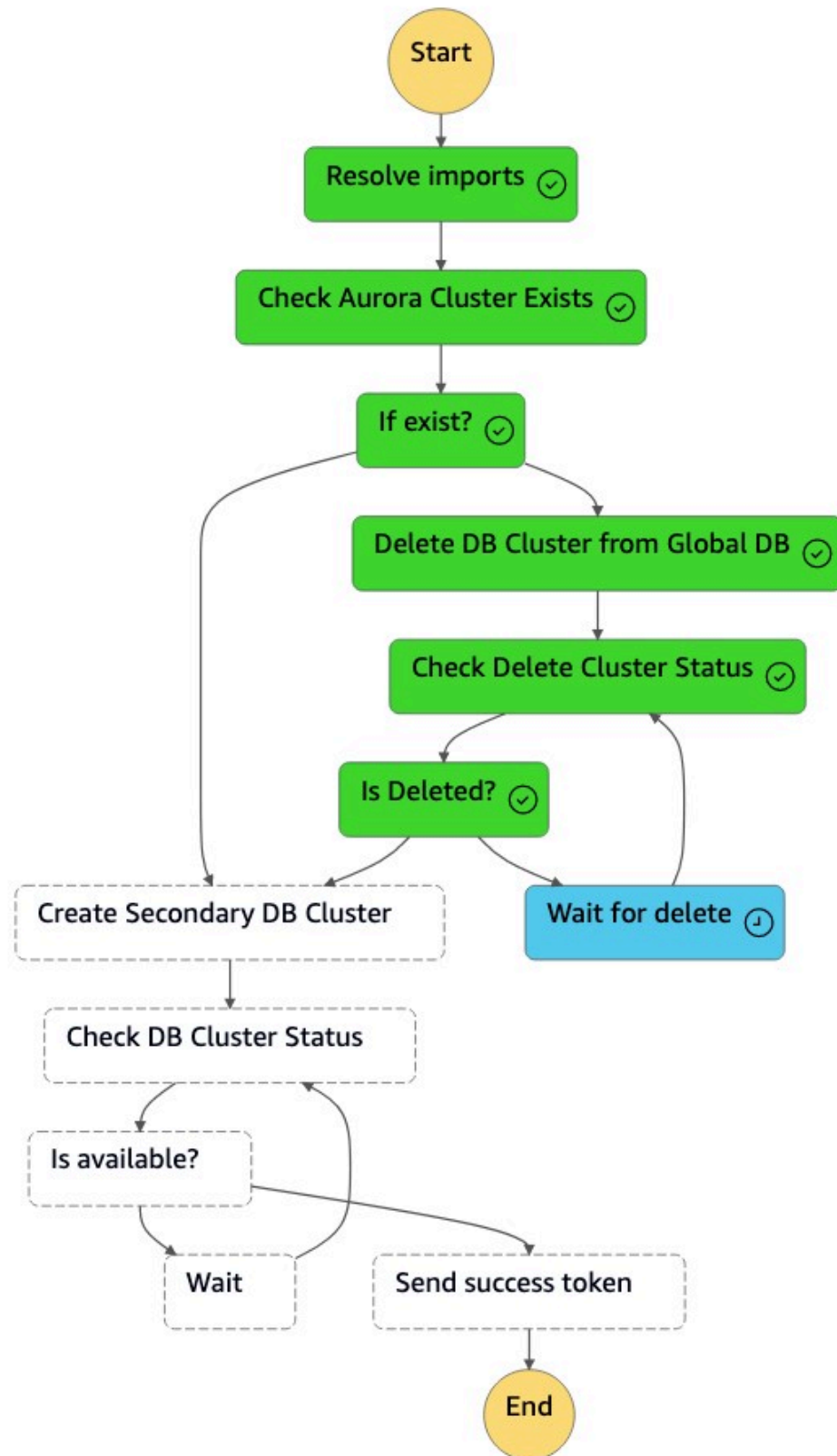
        "DBInstanceIdentifier": "!Import dr-globaldb-cluster-mysql-instance-
identifier",
        "Port": "!Import dr-globaldb-cluster-mysql-port",
        "DBInstanceClass": "!Import dr-globaldb-cluster-mysql-instance-class",
        "DBSubnetGroupName": "!Import dr-globaldb-cluster-mysql-subnet-group-
name",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

```

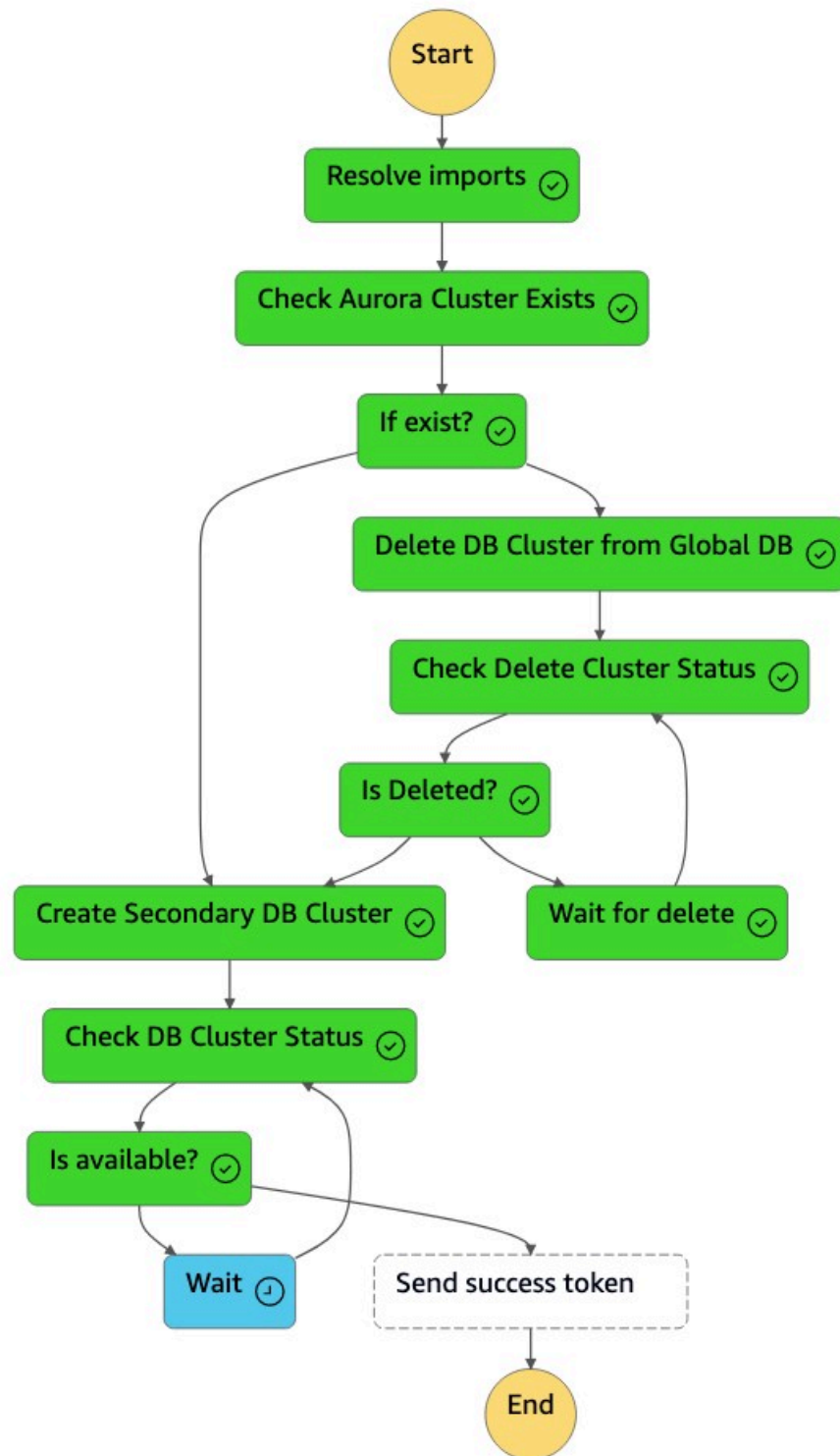
6. A máquina de DR Orchestrator FAILBACK estado lê o tipo de recurso como `CreateAuroraSecondaryDBCluster` e chama a máquina de `dr-orchestrator-stepfunction-create-Aurora-Secondary-cluster` estado.



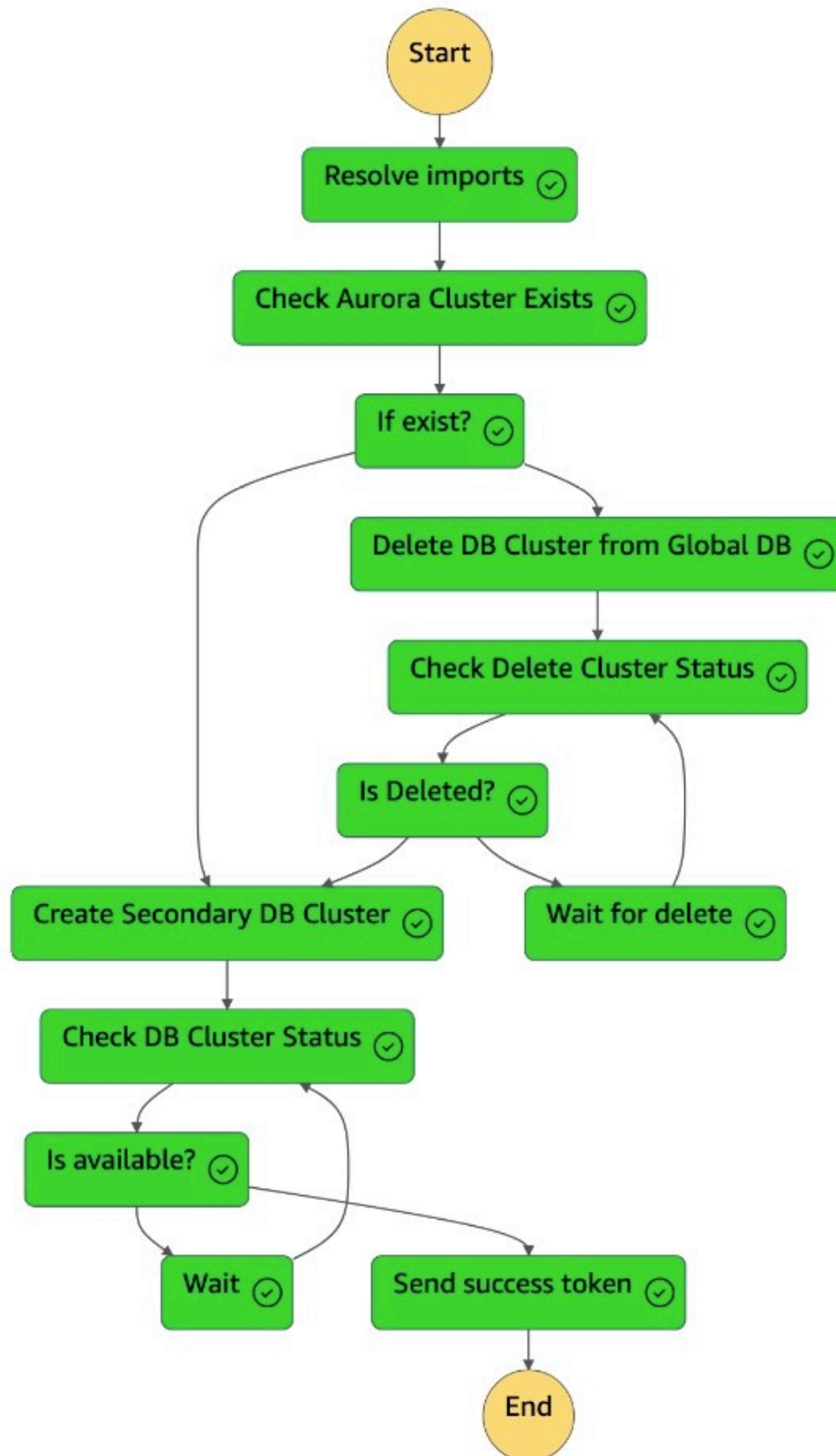
7. A máquina de `dr-orchestrator-stepfunction-create-Aurora-Secondary-cluster` estado exclui o cluster existente (`dbcluster-01`) da antiga região primária (`us-east-1`).



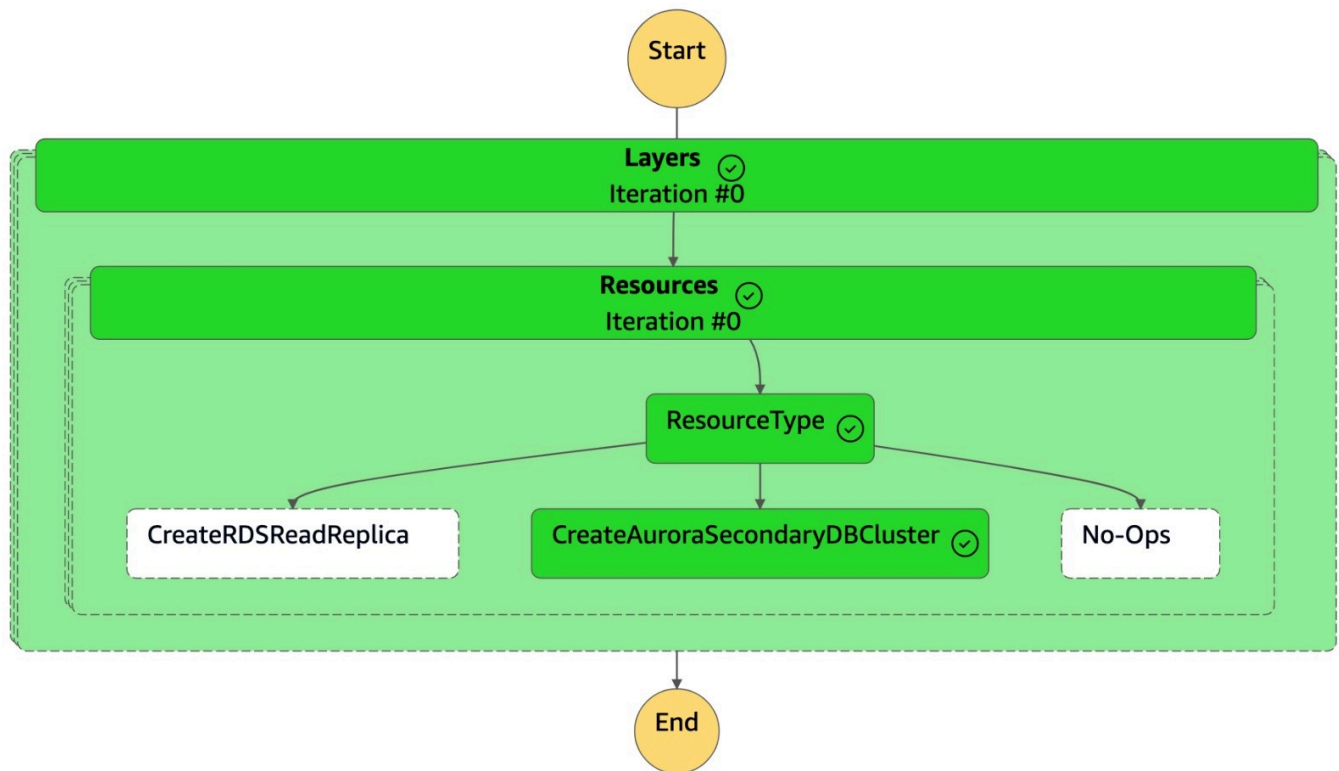
8. Depois que o cluster (`dbcluster-01`) é excluído, a máquina de estado cria um novo cluster (`dbcluster-01`) junto com a instância de banco de dados e se junta ao banco de dados global Aurora como cluster secundário para atender cargas de trabalho somente para leitura.



9. Depois que o cluster secundário estiver disponível, a máquina de `dr-orchestrator-stepfunction-create-Aurora-Secondary-cluster` estado será concluída e enviará um token de sucesso de volta para a máquina de DR Orchestrator Failback estado.



10A máquina de dr-orchestrator-stepfunction-FAILBACK estado está concluída.



11.Você pode verificar o banco de dados global do Aurora no console do Amazon RDS.

[Se você quiser realocar o cluster de banco de dados primário para us-east-1, siga as etapas mencionadas na seção Switchover.](#)

Implante a solução

Depois de configurar os pré-requisitos, siga as instruções no padrão do [DR Orchestrator](#) Framework. As instruções estão incluídas para a Amazon RDS Aurora e a Amazon ElastiCache. Para testar a solução, analise os pré-requisitos e as limitações e implante bancos de dados de exemplo em AWS.

Pré-requisitos

- Um ativo [Conta da AWS](#) para implantar essa solução.
- Permissões para criar buckets do [Amazon Simple Storage Service \(Amazon S3\)](#) e os recursos usados nos modelos. [AWS CloudFormation](#)
- Antes de executar a máquina de DR Orchestrator, coloque o estado em `FAILBACK`:
 - [Modifique manualmente a RDS instância](#) ou [modifique o cluster de banco de dados Amazon Aurora](#) e desative `DeletionProtection`.
 - Faça um [DB snapshot](#) antes de excluir a RDS instância da Amazon ou o cluster Aurora se você planeja realizar a análise da causa raiz (RCA).

Limitações

- A disponibilidade e o suporte dos recursos variam entre as versões específicas de cada mecanismo de banco de dados e as Regiões da AWS. Para obter mais informações sobre a disponibilidade de recursos e regiões para replicação entre regiões, consulte [Réplicas de leitura entre regiões](#).
- Essa solução foi testada usando uma instância de banco de dados primária com uma réplica de leitura. Se você quiser usar mais de uma réplica de leitura, teste a solução minuciosamente antes de implementá-la em um ambiente de produção.
- Para a Amazon RDS, a solução suporta somente os seguintes mecanismos: [Amazon RDS for MySQL](#), [Amazon RDS for PostgreSQL](#) e [Amazon RDS for MariaDB](#).

Implante bancos de dados de exemplo em AWS

Para demonstrar a solução de DR entre regiões para bancos de dados em AWS, você pode implantar o [Amazon RDS for MySQL](#), o [Amazon Aurora SQL MySQL-Compatible Edition](#) e o [ElastiCache Amazon \(OSSRedis\)](#) - Global Datastore. Use as instruções no App-Stack README.

arquivo do GitHub repositório [aws-cross-region-dr-databases](https://github.com/aws-samples/aws-cross-region-dr-databases). Neste exemplo, a região primária é `us-east-1` e a região secundária é `us-west-2`.

A pilha de aplicativos usa a seção AWS CloudFormation [Saídas](#) para exportar os parâmetros usados pelo DR Orchestrator Framework para automatizar o failover e o failback dos bancos de dados. AWS

Perguntas frequentes

Encontre respostas para perguntas sobre o DR Orchestrator Framework para failover e failback.

Quais são o RPO e o RTO que posso alcançar usando essa abordagem?

Para obter informações sobre objetivo de tempo de recuperação (RTO) e objetivo de ponto de recuperação (RPO), consulte a estratégia [Estratégia de recuperação de desastres para bancos de dados em AWS](#).

É obrigatório usar variáveis de AWS CloudFormation exportação?

Não, você pode transmitir diretamente o valor do banco de dados global Amazon Aurora ou da instância de banco de dados Amazon RDS diretamente no formato JSON (por exemplo:).

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

Posso usar o fluxo de trabalho de FAILOVER do DR Orchestrator para fazer failover em mais de um banco de dados? AWS

Sim, você pode passar mais de um recurso no arquivo de parâmetros de entrada para fazer failover em mais de um AWS banco de dados. O exemplo de código a seguir mostra o failover de uma réplica de leitura do Amazon RDS for MySQL e de um armazenamento de dados global ElastiCache Amazon (Redis OSS) em 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"
    }
  }
]
}
```

Como posso evitar o InvalidParameterCombination erro ao executar a máquina de estado FAILBACK do DR Orchestrator para o Amazon RDS?

O texto completo do erro é:

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

Para evitar o erro, [modifique a instância do RDS](#) desativando DeletionProtection antes de executar a máquina de DR Orchestrator FAILBACK estado.

Próximas etapas

Você pode usar o DR Orchestrator Framework para adicionar um processo de aprovação antes de iniciar as atividades de failover ou failback. Por exemplo, ao iniciar o failover durante um evento de DR, você pode adicionar uma máquina de estado na parte superior DR Orchestrator Failover para enviar uma [notificação do Amazon Simple Notification Service \(Amazon SNS\)](#) por e-mail. Assim que a aprovação for concedida, a atividade de failover será iniciada.

O failover de aplicativos também pode ser integrado ao DR Orchestrator Framework para alternar os aplicativos de entrada e ponto do DNS para o novo endpoint da instância ou cluster do banco de dados.

Você pode usar o [AWS X-Ray](#) rastreamento para obter a duração do failover para calcular o RTO. Você pode criar um painel de monitoramento com base no X-Ray.

Recursos

- [Replicação Regiões da AWS usando armazenamentos de dados globais \(documentação da Amazon ElastiCache \(Redis OSS\)\)](#)
- [Usar o failover em um Amazon Aurora Global Database](#) (documentação do Aurora)
- [Automatize o gerenciamento de endpoints do Amazon Aurora Global Database para failover planejado e não planejado](#) (publicação no blog)
- [Recuperação de desastres: 3 cenários de failover para seu banco de dados global Amazon Aurora com o Terraform \(Parte 2\) \(Postagem no blog\)](#)
- [Estratégia de recuperação de desastres para bancos de dados em AWS](#) (estratégia de orientação AWS prescritiva)
- [Automatize o failover e o failback entre regiões usando o DR Orchestrator Framework](#) (padrão de orientação prescritiva)AWS

Histórico do documento

A tabela a seguir descreve alterações significativas feitas neste guia. Se desejar receber notificações sobre futuras atualizações, inscreva-se em um [feed RSS](#).

Alteração	Descrição	Data
Publicação inicial	—	3 de maio de 2024

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.