



のリレーショナルデータベースの DR ソリューションを自動化する AWS

AWS 規範ガイドンス



AWS 規範ガイド: のリレーショナルデータベースの DR ソリューションを自動化する AWS

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

Amazon の商標およびトレードドレスは Amazon 以外の製品およびサービスに使用することはできません。また、お客様に誤解を与える可能性がある形式で、または Amazon の信用を損なう形式で使用することもできません。Amazon が所有していないその他のすべての商標は Amazon との提携、関連、支援関係の有無にかかわらず、それら該当する所有者の資産です。

Table of Contents

序章	1
目的	1
概要	3
フェイルオーバープロセス	4
DR Orchestrator FAILOVER アーキテクチャ	4
フェイルバックプロセス	5
DR Orchestrator FAILBACK アーキテクチャ	5
DR オーケストレーターのフェイルオーバー	7
パラメーターの詳細	8
DR オーケストレーターの失敗	9
Aurora ステートマシンと Step Functions ステートマシン	12
定常状態	12
イベント状態	12
切り替え	13
手動の計画外のフェイルオーバー	18
フェイルバック	22
ソリューションをデプロイする	31
よくある質問	33
このアプローチを使用して達成できる RPO と RTO は何ですか?	33
AWS CloudFormation エクスポート変数の使用は必須ですか?	33
DR オーケストレーターの FAILOVER ワークフローを使用して、複数の AWS データベースを フェイルオーバーできますか?	33
Amazon RDS の DR オーケストレーター FAILBACK ステートマシンを実行するときに InvalidParameterCombination エラーを回避するにはどうすればよいですか?	34
次のステップ	35
リソース	36
ドキュメント履歴	37
.....	xxxviii

のリレーショナルデータベースの DR ソリューションを自動化する AWS

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

2024 年 5 月 ([ドキュメント履歴](#))

アマゾン ウェブ サービス (AWS) で[ディザスタリカバリ \(DR\) 戦略](#)を計画する場合、組織内のデータベースに DR ソリューションを実装できます。データベースインスタンスのフェイルオーバーを実行するようにイベント駆動型アーキテクチャを設定することで、DR ソリューションを自動化できます。このアプローチの利点は次のとおりです。

- 手動調整によるヒューマンエラーの削減
- 場合によっては、最短の目標復旧時間 (RTO) を提供する
- DR ソリューションの繰り返し可能なテストの促進

自動化がない場合、プライマリの停止により AWS リージョン、組織内のリソースはランブックのステップを手動で実行する必要があります。これらのステップには、新しいプライマリデータベースインスタンスとして機能する AWS リージョン 内にセカンダリで Amazon Relational Database Service (Amazon RDS) リードレプリカを昇格させるなど、複数のプロセスが含まれる場合があります。は、これらのアクションを自動化するための [AWS SDK for Python \(Boto3\)](#) APIs AWS を提供し、組織のニーズに合わせた DR ソリューションの構成要素を形成することができます。

このガイドでは、一元化されたマニフェスト駆動型 DR オーケストレーターについて説明します。このオーケストレーターは、同じ AWS リージョン 内の全体で AWS データベースの DR ソリューションを実現するのに役立ちます AWS アカウント。フェイルオーバーを実行する決定は、依然として人間による介入が必要であり、自動化の対象外です。

このガイドの対象者は、アプリケーションアーキテクト、インフラストラクチャアーキテクト、データベースアーキテクト、データベースコンサルタントです。

目的

DR Orchestrator Framework ソリューションを実装すると、次の成果を達成できます。

- 停止中の手動エラーを減らす – DR プロセスを手動で実行すると、エラーが発生しやすくなります。DR アクティビティのフェーズ中にアクションを自動化すると、手動エラーが軽減され、フェイルオーバーやフェイルバックがスムーズになります。
- 大規模な DR の管理 – 組織に大量のデータベースがある場合、DR の自動化を構築することで DR ソリューションをスケールできます。DR Orchestrator Framework を使用して、AWS データベース DR ソリューションを自動化できます。DR Orchestrator は、JSON 形式のパラメータファイルを使用して、AWS データベースのフェイルオーバーまたはフェイルバックのステップを自動化します。

DR オーケストレーターフレームワークの概要

DR オーケストレーターフレームワークは、AWS データベースのクロスリージョン DR をオーケストレーションおよび自動化するワンクリックソリューションを提供します。フェイルオーバー [AWS Step Functions](#) とフェイルバック中に必要なステップを実行する [AWS Lambda](#) には、とを使用します。Step Functions ステートマシンは、オーケストレーター設計内での意思決定の基礎を提供します。フェイルオーバーまたはフェイルバックアクションを実行するための API オペレーションは、ステートマシン内から呼び出される Lambda 関数にコードされます。Lambda 関数は [AWS SDK for Python \(Boto3\)](#) API を実行して AWS データベースとやり取りします。APIs

DR Orchestrator Framework には、フェイルオーバーフェーズとフェイルバックフェーズに対応する 2 つのメインステートマシンが含まれています。

Amazon RDS の場合、フェイルオーバーフェーズはクロスリージョン RDS リードレプリカをスタンバイオン DB インスタンスに昇格させます。Amazon Aurora の場合、まれに予期しない停止時にプライマリリージョンが停止すると、そのライターノードは使用できません。ライターノードとセカンダリクラスター間のレプリケーションが停止します。グローバルデータベースからセカンダリクラスターをデタッチし、スタンバイオンクラスターとして昇格させる必要があります。アプリケーションは、スタンバイオンクラスターに接続して書き込みトラフィックを送信できます。この同じプロセスを使用して、グローバルデータベースのプライマリ DB クラスターをセカンダリリージョンに切り替えることができます。このアプローチは、次のような制御されたシナリオで使われます。

- 運用メンテナンス
- 計画された運用手順
- 新しいプライマリクラスターとしての Amazon ElastiCache (Redis OSS) セカンダリクラスターの昇格

フェイルバックフェーズでは、ライブプライマリリージョンと新しいセカンダリリージョンの間でデータのライブレプリケーションを確立します。

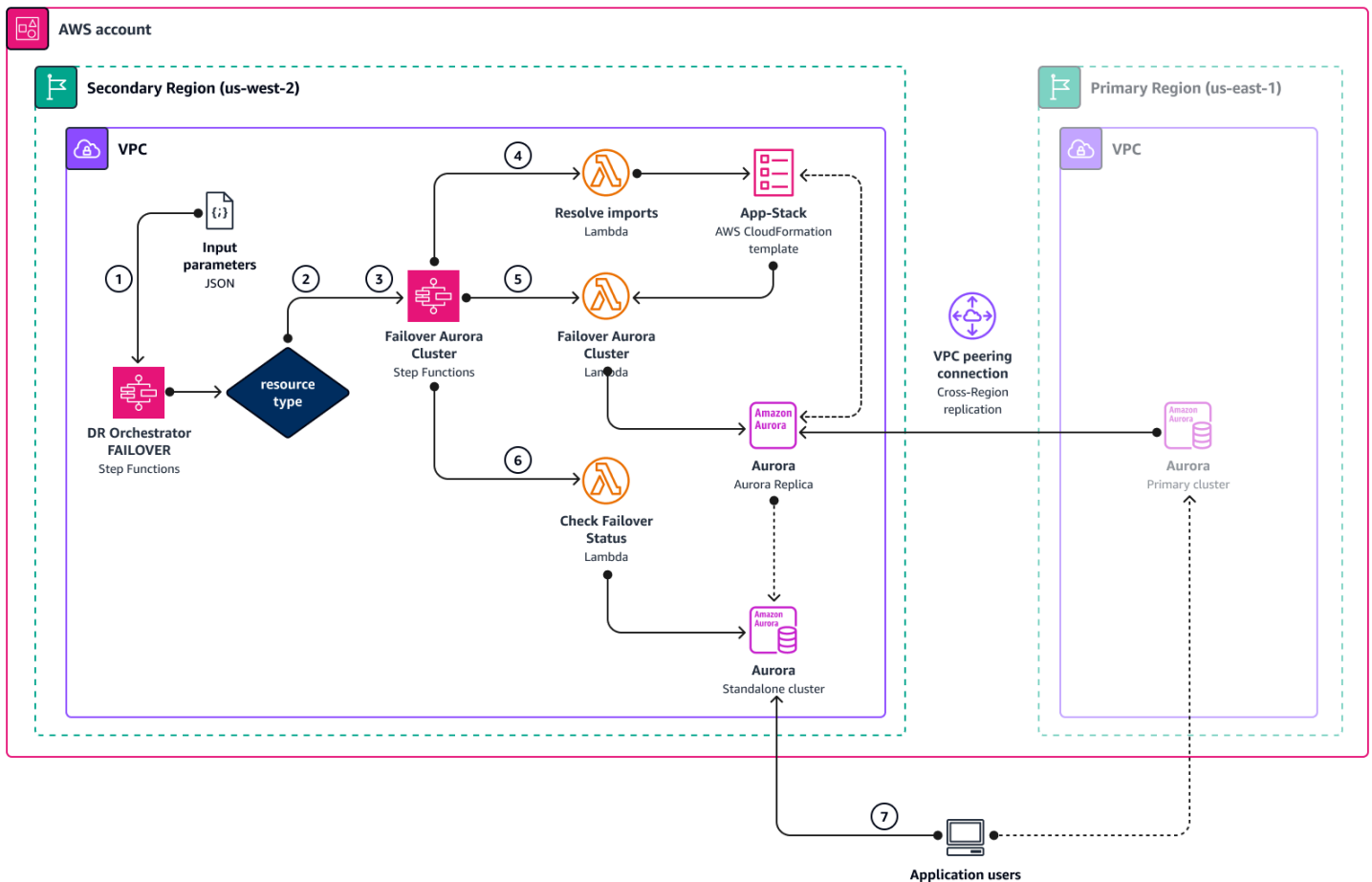
DR Orchestrator はデータベースにのみ適用されることを理解することが重要です。これらのデータベースを参照し、同じリージョンにあるすべてのアプリケーションには、個別のタンドムフェイルオーバーソリューションが必要になる場合があります。データベースがセカンダリリージョンにフェイルオーバーされたら、アプリケーションを更新して、データソースとして機能する新しいデータベースインスタンスに接続する必要があります。

フェイルオーバープロセス

フェイルオーバーを実行するには、DR Orchestrator FAILOVERステートマシンを実行します。この段階では、セカンダリデータベースは、リードレプリカ (Amazon RDS) またはセカンダリクラスター (Amazon Aurora) として、セカンダリリージョンに既に存在します。DR Orchestrator FAILOVER ステートマシンを実行すると、セカンダリデータベースがプライマリに昇格します。

DR Orchestrator FAILOVER アーキテクチャ

次の図は、DR Orchestrator を使用する場合の Amazon Aurora のフェイルオーバープロセスの概念を示しています。Amazon Aurora と Amazon ElastiCache は同じワークフローを使用しますが、ステートマシンと Lambda 関数は異なります。



1. DR Orchestrator FAILOVER ステートマシンは入力 JSON パラメータを読み取ります。
2. resourceType パラメータに基づいて、ステートマシンは Promote RDS Read Replica、Failover Aurora Cluster または の他の ステートマシンを呼び出しま

- すFailover ElastiCache。入力で複数のリソースが渡されると、これらのステートマシンは並行して実行されます。
3. Failover Aurora Cluster ステートマシンは、次の 3 つのステップのそれぞれで Lambda 関数を呼び出します。
 4. Resolve imports Lambda 関数は、App-Stack AWS CloudFormation テンプレートの実際の値 `! import <export-variable-name>` で解決されます。
 5. Lambda Failover Aurora Cluster 関数は、リードレプリカをスタンドアロン DB インスタンスとして 昇格させます。
 6. Lambda Check Failover Status関数は、昇格された DB インスタンスのステータスをチェックします。ステータスが AVAILABLE になると、Lambda 関数は成功トークンを呼び出し元のステートマシンに送り返し、完了します。
 7. プライマリデータベースになった DR リージョン (us-west-2) のスタンドアロンデータベースにアプリケーションをリダイレクトできます。

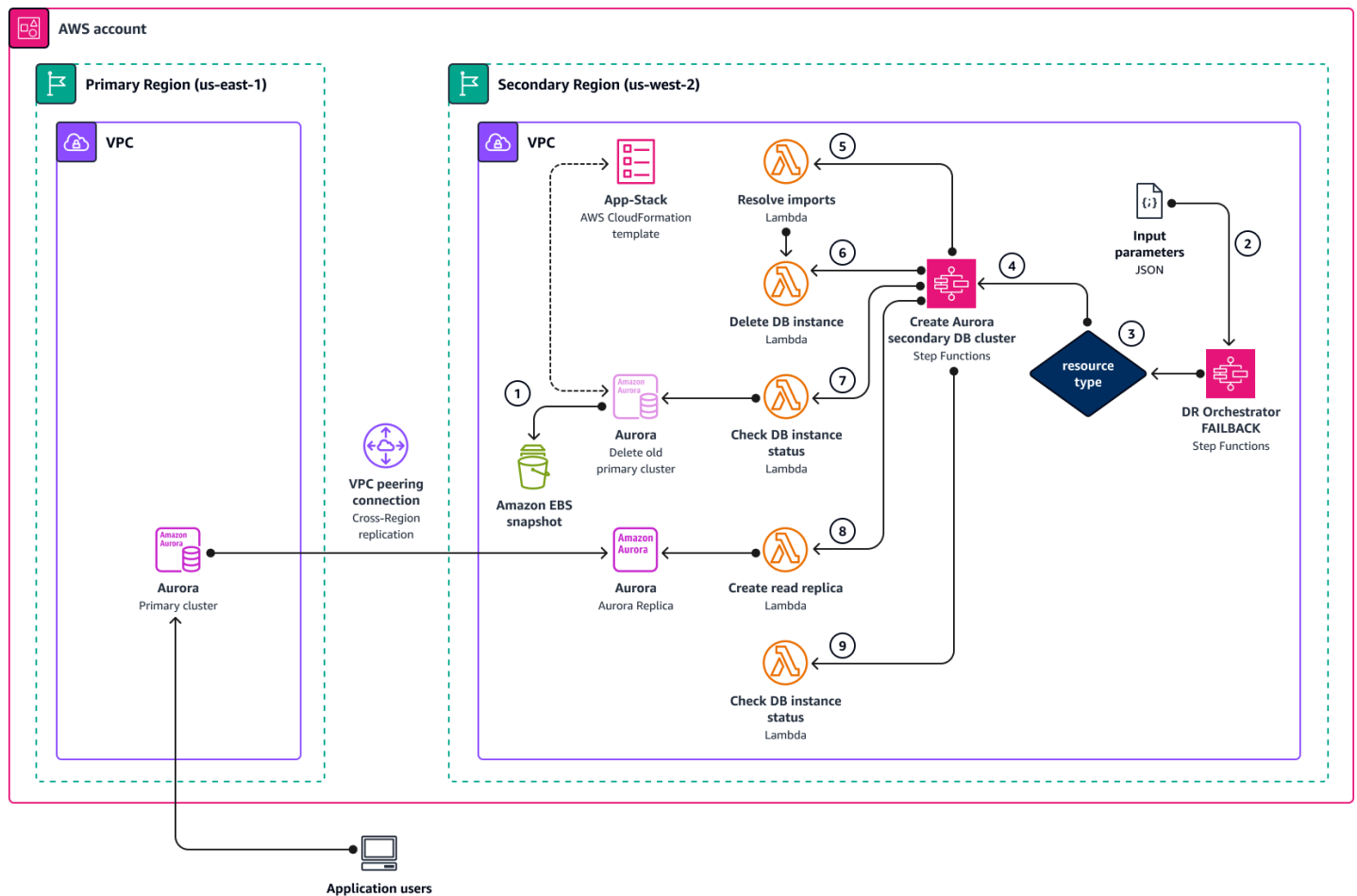
フェイルバックプロセス

以前のプライマリリージョン (us-east-1) が再び起動したら、そのリージョンにフェイルバックして、のデータベースが再びプライマリus-east-1になります。フェイルバックを開始するには、DR Orchestrator FAILBACKステートマシンを実行します。名前が示すように、このステートマシンは新しいプライマリリージョン (us-west-2) の変更を、現在のセカンダリとして機能する以前のプライマリリージョン (us-east-1) にレプリケートし始めます。

2 つのリージョン間でレプリケーションが確立されたら、フェイルバックを開始できます。フェイルバックして元のプライマリリージョン (us-east-1) に戻るには、現在のセカンダリリージョン (us-east-1) でDR Orchestrator FAILOVERステートマシンを実行してプライマリリージョンに昇格します。

DR Orchestrator FAILBACK アーキテクチャ

次の図は、DR Orchestrator を使用する場合の Amazon Aurora のフェイルバックプロセスの概念を示しています。



1. フェイルバックを開始する前に、根本原因分析 (RCA) を実行するときに使用する手動 DB スナップショットを作成します。

また、前のプライマリリージョン () で Aurora クラスター DeletionProtection の を無効にします us-east-1。

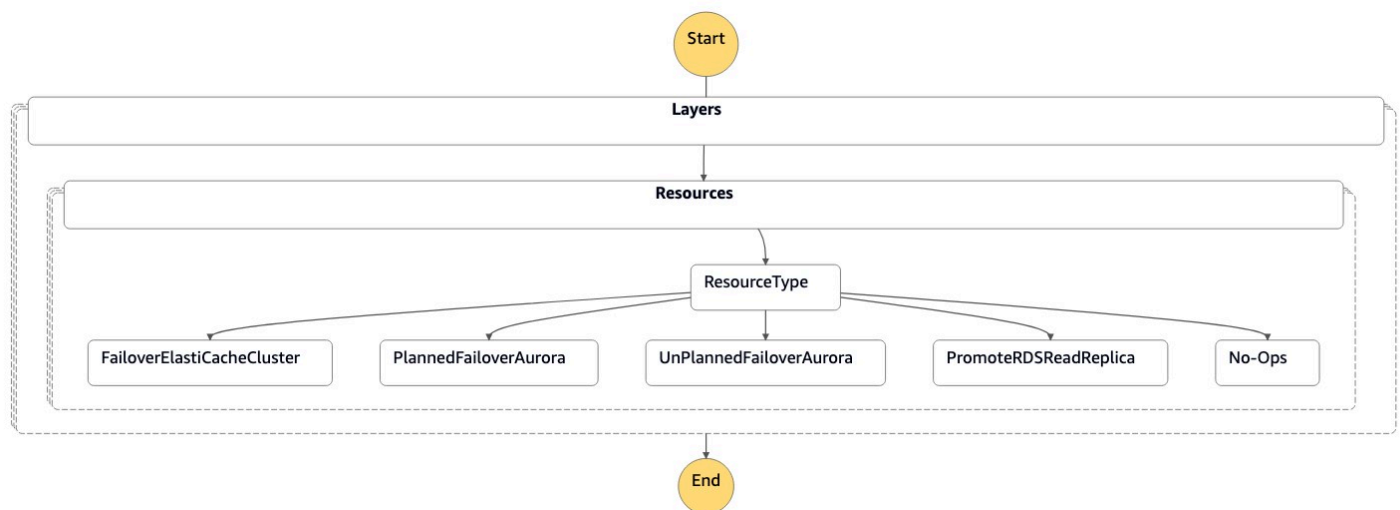
2. DR Orchestrator FAILBACK ステートマシンは入力 JSON パラメータを読み取ります。
3. に基づいて resourceType、DR Orchestrator FAILBACK ステートマシンは Create Aurora Secondary DB Cluster ステートマシンを呼び出します。
4. Create Aurora Secondary DB Cluster ステートマシンは、次の 5 つのステップのそれぞれで Lambda 関数を呼び出します。
5. Resolve import Lambda 関数は、App-Stack CloudFormation テンプレートの実際の値 "import <export-variable-name>" で解決されます。
6. Delete DB Instance Lambda 関数は、以前のプライマリインスタンスを削除します。

7. Check DB instance status Lambda 関数は、DB が削除され Delete DB Instance status まで をチェックします。
8. Create Read Replica Lambda 関数は、新しいプライマリリージョンにある DB インスタンス からセカンダリリージョンにリードレプリカを作成します。
9. Check DB instance status Lambda 関数は、リードレプリカ DB インスタンスのステータス をチェックします。ステータスが AVAILABLE の場合、Lambda 関数は成功トークンを呼び出し元のステートマシンに送り返し、完了します。

DR オーケストレーターのフェイルオーバー

プライマリリージョン (us-east-1) がダウンしている場合、または運用メンテナンスなどの計画されたイベント中に、DR イベントで DR Orchestrator FAILOVER ステートマシンを使用します。

関数を呼び出して、単一または複数のデータベースを並行してフェイルオーバーできます。



ステートマシンは、次のコードに示すように、JSON 形式のパラメータを受け入れます。

```

{
  "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"
        }
    }
]
```

パラメーターの詳細

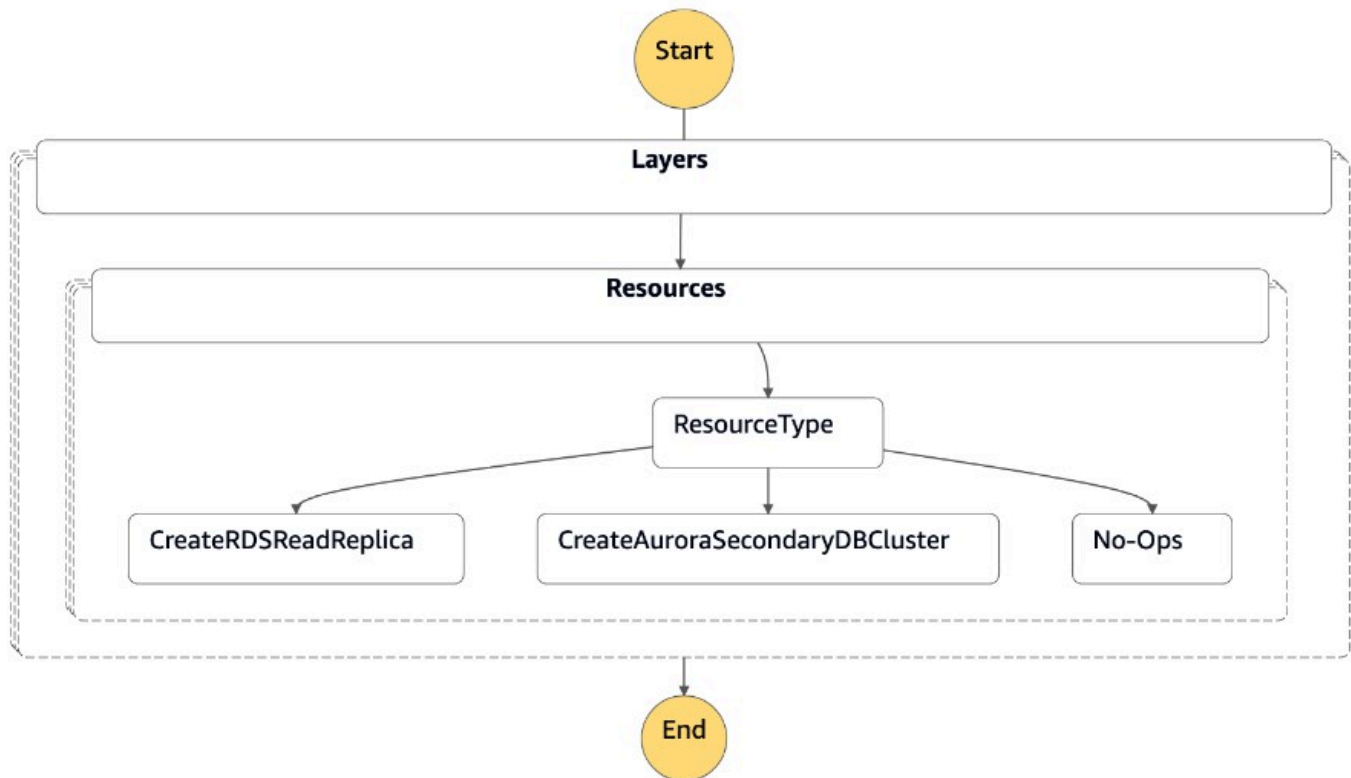
次の表は、DR Orchestrator FAILOVERステートマシンで使用されるパラメータを示しています。

パラメータ名	説明	予想される値
layer (必須: 数値)	処理シーケンス。レイヤー 1 で定義されたすべてのリソースは、レイヤー 2 リソースを実行する前に実行する必要があります。	1 または 2 など
リソース (必須: デクシヨナリの配列)	1 つのレイヤー内のすべてのリソースが並行して実行されます。	<pre>{ "resourceType": " String", "resourceName": "St ring", "parameters": { "<param1>": "<!Import cft-output-1"></pre>

パラメータ名	説明	予想される値
		<pre>.... }</pre>
resourceType (必須: 文字列)	リソースを識別するリソースのタイプ	PromoteRDSReadReplica、または FailoverElasticCacheCluster
resourceName (オプション: 文字列)	これらのリソースが属するアプリケーションポートフォリオを特定するには	Promote RDS for MySQL Read Replica
パラメータ (必須: ディクショナリの配列)	データベースの AWS フェイルオーバーまたはフェイルバックに必要なパラメータのリスト	<pre>{ "<param1>": "<!Import cft- output-1>", "<param2>": "<!Import cft- output-2>", }</pre>

DR オーケストレーターの失敗

以前のプライマリリージョン (us-east-1) が稼働している場合、DR イベントの後に DR Orchestrator FAILBACKステートマシンを使用します。以前のプライマリリージョンの Amazon RDS [のリードレプリカ](#) を新しいプライマリリージョン (us-west-2) から作成して、DR 戦略に準拠させることができます。これは計画されたイベントであるため、このアクティビティを週末またはオフピークの営業時間に、推定ダウンタイムでスケジュールできます。



ステートマシンは、次のコードに示すように、JSON 形式のパラメータを受け入れます。

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

Aurora ステートマシンと Step Functions ステートマシン

このセクションでは、Amazon Aurora クラスターのフェイルオーバーとフェイルバックに固有のプロセスとステートマシンについて説明します。クラスターはグローバルデータベースとして設定されます。

Note

デモンストレーションの目的で、この例では Aurora MySQL 互換エディションを使用しています。Aurora PostgreSQL 互換エディションでも同様の手順を使用できます。

定常状態

定常状態では、Amazon Aurora MySQL 互換グローバルデータベース (dr-globaldb-cluster-mysql) が 2 つの DB クラスターで作成されています。最初の DB クラスター (db-cluster-01) は、読み取り/書き込みワークロードを処理するためにプライマリ AWS リージョン (us-east-1) に作成されています。2 番目の DB クラスター (db-cluster-02) は、読み取り専用ワークロードをサーバーするためにセカンダリリージョン (us-west-2) に作成されています。

DR ソリューションを提供することに加えて、アプリケーションからセカンダリ DB クラスターに読み取りクエリをルーティングすることで、プライマリ DB クラスターの負荷を軽減できます。これらの各クラスターには dbcluster-02-usw2-instance-2、それぞれ dbcluster-01-use1-instance-1 および という名前のデータベースインスタンスが 1 つ含まれています。

イベント状態

Amazon Aurora グローバルデータベースを使用すると、災害の計画と復旧をかなり迅速に行うことができます。災害からの復旧は通常、目標復旧時間 (RTO) と目標復旧時点 (RPO) の値を使用して測定されます。詳細については、[「Amazon Aurora グローバルデータベースでのスイッチオーバーまたはフェイルオーバーの使用」](#)を参照してください。

Aurora グローバルデータベースでは、フェイルオーバーには 2 つの異なるアプローチがあります。

- スwitchオーバー (マネージド計画フェイルオーバー)
- フェイルオーバー (手動の計画外のフェイルオーバー、またはデタッチと昇格)

切り替え

スイッチオーバーは、運用メンテナンスやその他の計画された運用手順など、制御された環境を対象としています。マネージド計画フェイルオーバーを使用すると、Aurora グローバルデータベースのプライマリ DB クラスターをセカンダリリージョンの 1 つに再配置できます。スイッチオーバーはセカンダリ DB クラスターがプライマリデータベースと同期されるまで待機するため、RPO は 0 (データ損失なし) です。詳細については、[「Amazon Aurora グローバルデータベースのスイッチオーバーの実行」](#)を参照してください。

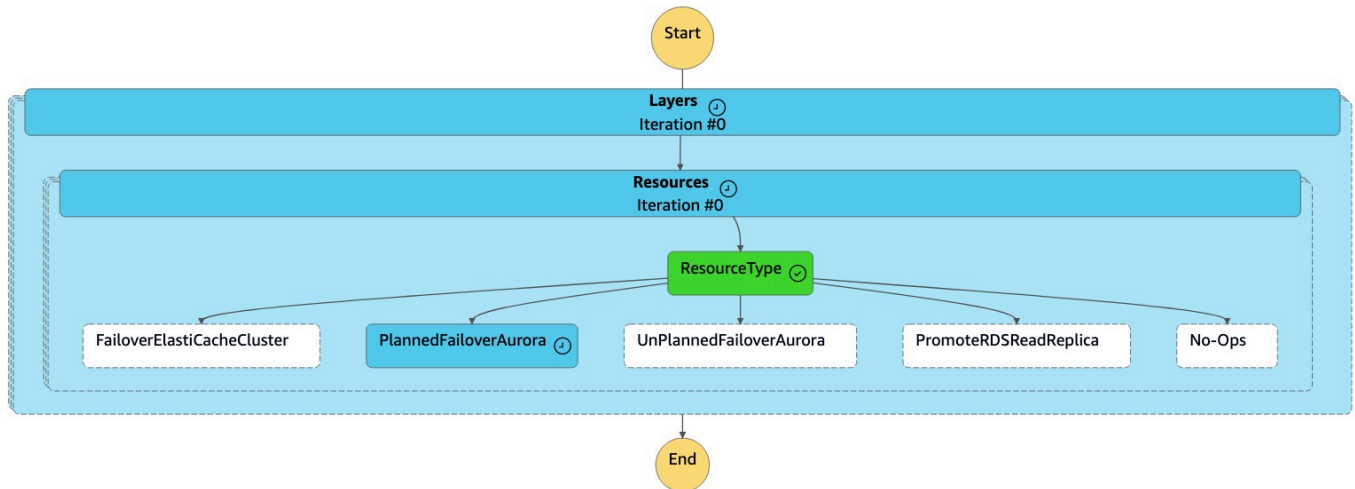
dr-orchestrator-stepfunction-FAILOVER ステートマシンはイベント状態中に呼び出され、プライマリクラスターを選択したセカンダリリージョン () に切り替えますus-west-2。

スイッチオーバーを実行するには、以下を実行します。

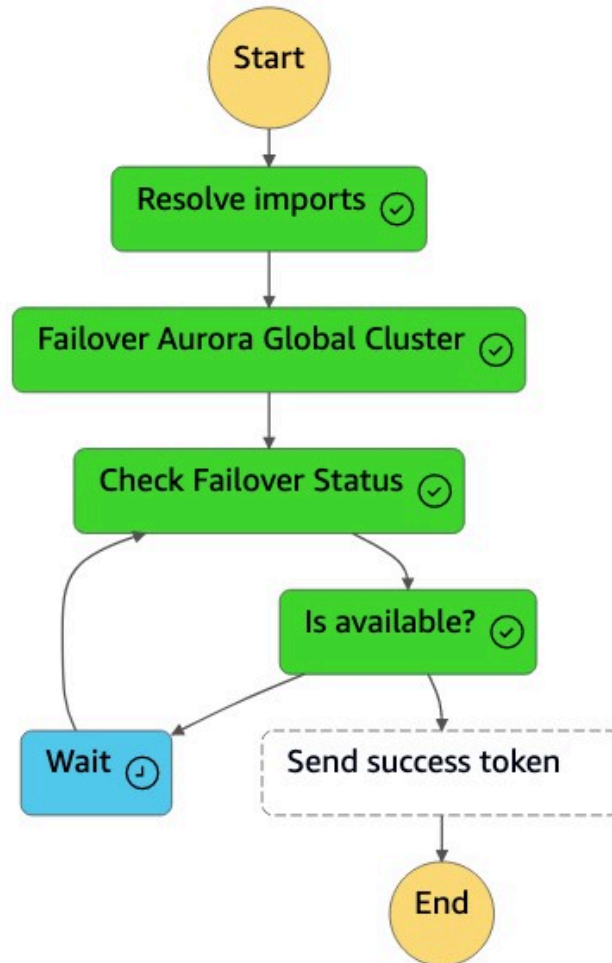
1. AWS マネジメントコンソールにサインインします。
2. リージョンを DR リージョン () に変更しますus-west-2。
3. サービスに移動し、Step Functions を選択します。
4. dr-orchestrator-stepfunction-FAILOVER ステートマシンに移動します。
5. 実行の開始を選択し、Input - optionalセクションに次の JSON コードを入力します。

```
{
  "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. `dr-orchestrator-stepfunction-FAILOVER` ステートマシンはリソースタイプを `PlannedFailoverAuroraMySQL` として読み取り、`dr-orchestrator-stepfunction-planned-Aurora-failover` ステートマシンを呼び出して Aurora グローバルデータベースをフェイルオーバーします。



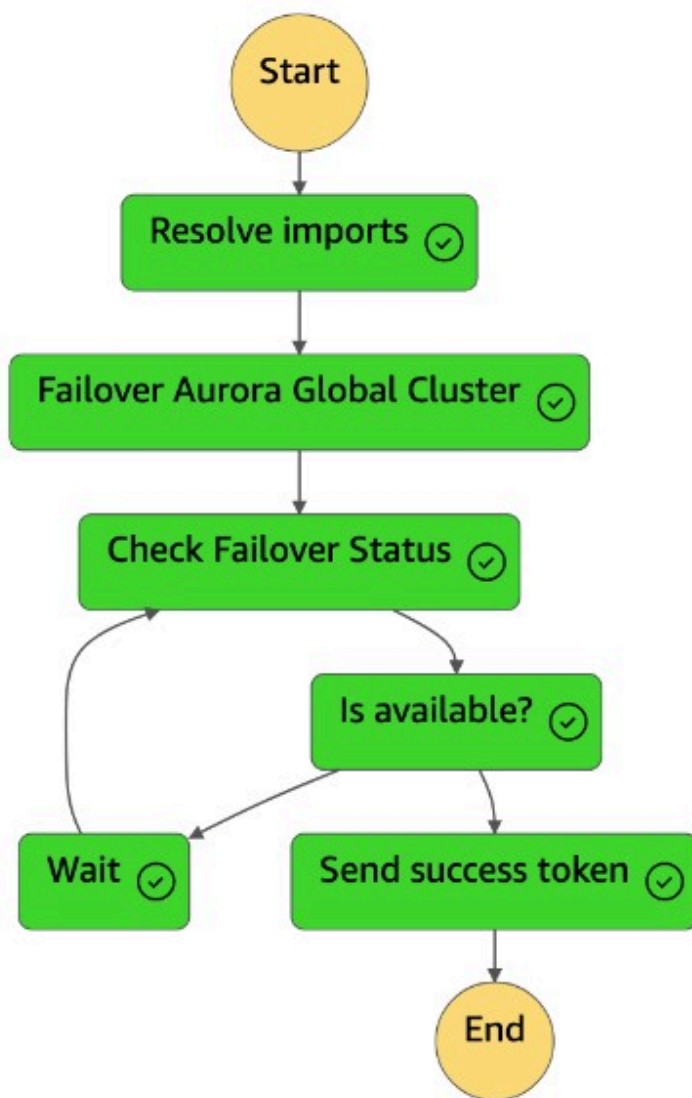
7. `dr-orchestrator-stepfunction-planned-Aurora-failover` ステートマシンは次の手順を実行して、Aurora MySQL 互換のグローバルデータベースロールを切り替えます。



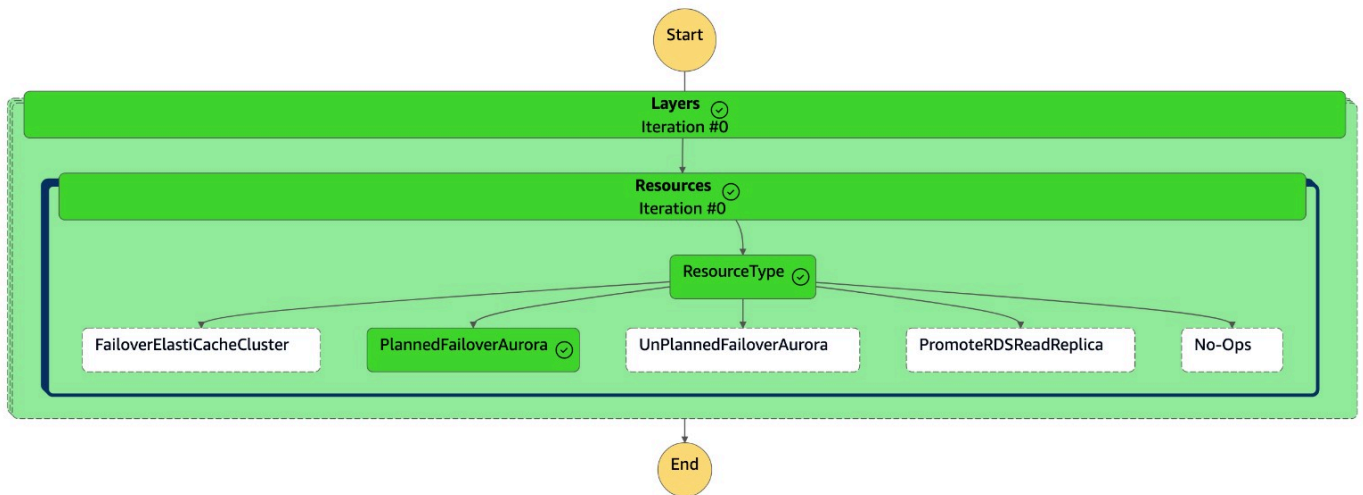
Step	説明	予想される値
インポートの解決	Lambda 関数は、!Import <variable name>値を実際の名前に置き換えます。	"!Import dr-global db-cluster-mysql-global-identifier" は に置き換えられます"dr-globaldb-cluster-mysql" 。

Step	説明	予想される値
フェイルオーバー Aurora グローバルクラスター	Lambda 関数は failover_global_cluster Boto3 APIs を呼び出して、Aurora グローバルデータベースをフェイルオーバーします。	<pre>{ 'GlobalCluster': { 'GlobalClusterIdentifier': 'dr-global-cluster-mysql', 'GlobalClusterResourceId': 'cluster-cce7f9bec2846db4', 'GlobalClusterArn': 'arn:aws:rds::xxx', 'Status': 'failing-over', } }</pre>
フェイルオーバーステータスを確認する	Lambda 関数は describe_db_clusters Boto3 APIs を呼び出して、フェイルオーバーのステータスを確認します。	変更中、使用可能
成功トークンを送信する	Lambda 関数は send_task_success Boto3 APIs を呼び出し、成功トークンを DR Orchestrator Failover ステートマシンに送信します。	h7X RiCdLtd/83p1E0dMccoxIzFhglsdkzpK9mBVKZsp7d9yrT1W

- Amazon RDS コンソールに移動します。Status では、Aurora グローバルデータベースの値が Available から Switching over または Modifying に変わります。
- dr-orchestrator-stepfunction-planned-Aurora-failover ステートマシンが完了すると、成功トークンを dr-orchestrator-stepfunction-FAILOVER ステートマシンに送り返します。



10dr-orchestrator-stepfunction-FAILOVER ステートマシンが完了しました。



コンソールでは、セカンダリクラスター (dbcluster-02) のロールがプライマリクラスターになり、クラスターは読み取り/書き込みワークロードを処理する準備が整いました。元のプライマリクラスター (dbcluster-01) のロールがセカンダリクラスターとしてリストされるようになりました。

手動の計画外のフェイルオーバー

まれに、Aurora グローバルデータベースのプライマリで予期しない停止が発生することがあります。AWS リージョン。この場合、プライマリ Aurora DB クラスターとその読み取りノードを使用できなくなり、プライマリクラスターとセカンダリクラスター間のレプリケーションが停止します。ダウンタイム (RTO) とデータ損失 (RPO) の両方を最小限に抑えるには、クロスリージョンフェイルオーバーを迅速に実行し、Aurora グローバルデータベースを再構築します。詳細については、「[Amazon Aurora グローバルデータベースを予期しない停止から復旧する](#)」を参照してください。

計画外のフェイルオーバーを実行するには、Aurora グローバルデータベースからセカンダリクラスターをデタッチする必要があります。計画外のフェイルオーバーを実行する前に、プライマリ Aurora DB クラスターでのアプリケーションの書き込みを停止します。フェイルオーバーが正常に完了したら、新しいプライマリ DB クラスターに書き込むようにアプリケーションを再設定します。このアプローチは、データ損失を防ぐのに役立ちます。また、フェイルオーバープロセス中にプライマリライターノードがオンラインに戻った場合も、データの不整合を回避できます。

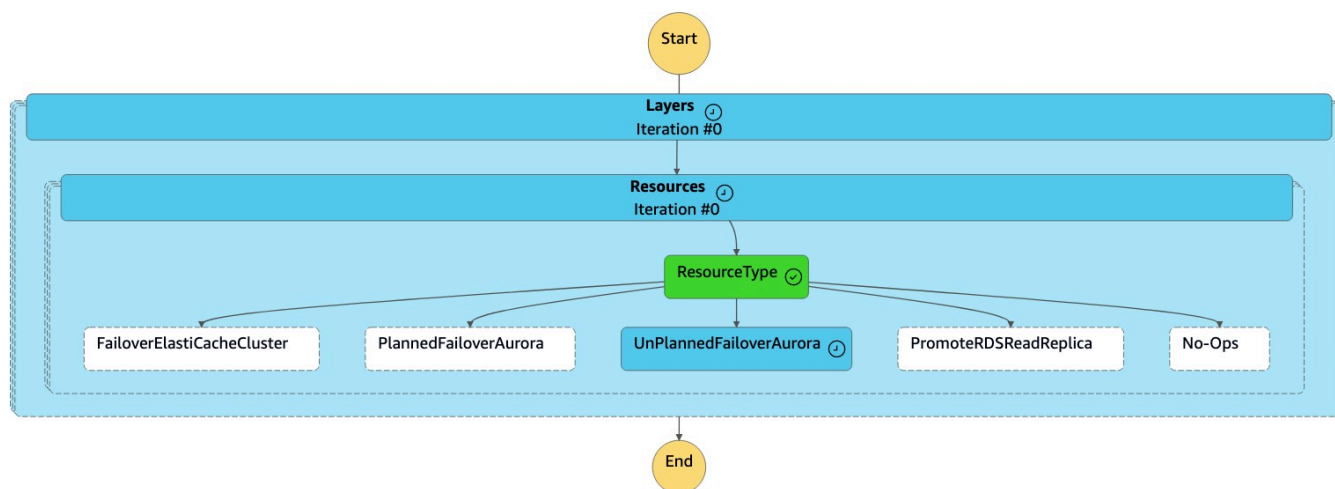
計画外のフェイルオーバーを実行するには、dr-orchestrator-stepfunction-FAILOVERステートマシンを呼び出します。この例では、セカンダリクラスター (db-cluster-02) は DR リージョン (us-west-2) の定常状態にあります。

フェイルオーバーを実行するには、以下を実行します。

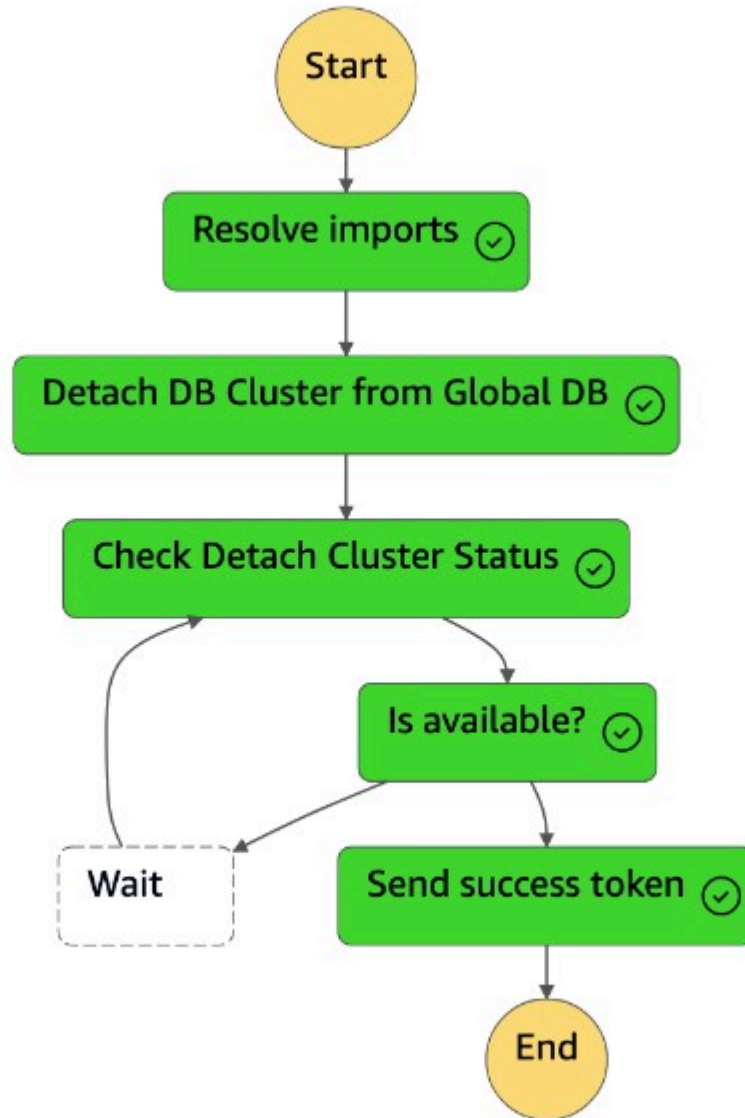
1. コンソール にサインインします。
2. リージョンを DR リージョン () に変更しますus-west-2。
3. サービスに移動し、Step Functions を選択します。
4. dr-orchestrator-stepfunction-FAILOVER ステートマシンに移動します。
5. 実行の開始を選択し、 を UnPlannedFailoverAuroraとして使用して、Input - optionalセクションに次の JSON コードを入力しますresourceType。

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

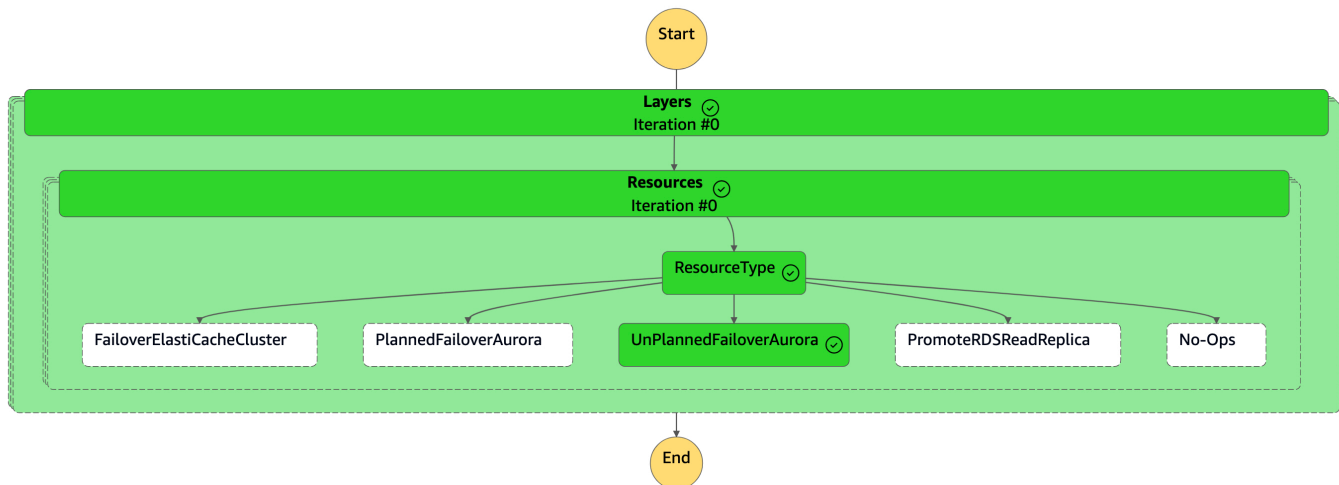
6. dr-orchestrator-stepfunction-FAILOVER ステートマシンはリソースタイプを として読み取りUnPlannedFailoverAuroraMySQL、 dr-orchestrator-stepfunction-unplanned-Aurora-failoverステートマシンDetach Cluster from Global Databaseからタスクを呼び出します。



7. Detach Cluster from Global Database タスクは、グローバルデータベースからセカンダリクラスターをデタッチ (削除) します。



8. セカンダリクラスター (dbcluster-02) はスタンドアロンクラスターに昇格し、読み取り/書き込みワークロードを処理できます。
9. dr-orchestrator-stepfunction-FAILOVER ステートマシンが完了しました。



10.セカンダリクラスター (dbcluster-02) は Aurora グローバルデータベースからデタッチされ、読み取り/ライターワークロードを処理するスタンドアロンクラスターになります。

11.新しいクラスターエンドポイントを使用して、この新しいスタンドアロン Aurora DB クラスターにすべての書き込みオペレーションを送信するようにアプリケーションを再設定します。

フェイルバック

フェイルバックは、災害 (またはスケジュールされたイベント) が解決された後、データベースを元の (または新しい) プライマリリージョンに返します。計画外の停止が解決したら、以前のプライマリリージョンを Aurora グローバルデータベースに戻したい場合があります。まず、以前のプライマリリージョンから既存の DB クラスターを削除し、新しいプライマリリージョンから新しい DB クラスターを作成し、マネージド計画フェイルオーバープロセスを使用して新しいクラスターのロールを切り替えます。

これは、オフピーク時または週末に実行できる計画されたアクティビティと見なすことができます。

で作成されたため、以前のプライマリリージョン (us-east-1) からDR Orchestrator FAILBACKステートマシンを実行するDeletionProtection 前に、[Amazon Aurora DB クラスターを手動で変更](#)し、を無効にする必要がありますDeletionProtection。

DR Orchestrator Framework はdr-orchestrator-stepfunction-FAILBACK ステートマシンを使用して、既存のクラスターを削除し、以前のプライマリリージョンに新しいクラスターを作成するステップを自動化します。

を無効にするにはDeletionProtection、以下を実行します。

1. コンソール にサインインします。
2. リージョンを以前のプライマリリージョン () に変更しますus-east-1。
3. Amazon RDS コンソールに移動し、クラスター名 (dbcluster-01) を選択し、変更を選択します。
4. 削除保護で、削除保護を有効にするチェックボックスをオフにし、続行を選択します。
5. すぐに適用を選択し、クラスターの変更を選択します。

DR Orchestrator FAILBACK ステートマシンは、以前のプライマリリージョン () からのフェイルバックプロセス中に呼び出されますus-east-1。

フェイルバックを実行するには、以下を実行します。

1. コンソール にサインインします。
2. リージョンを以前のプライマリリージョン () に変更しますus-east-1。
3. サービスに移動し、Step Functions を選択します。
4. DR Orchestrator FAILBACK ステートマシンに移動します。
5. 実行の開始を選択し、Input - optionalセクションに次の JSON コードを入力します。

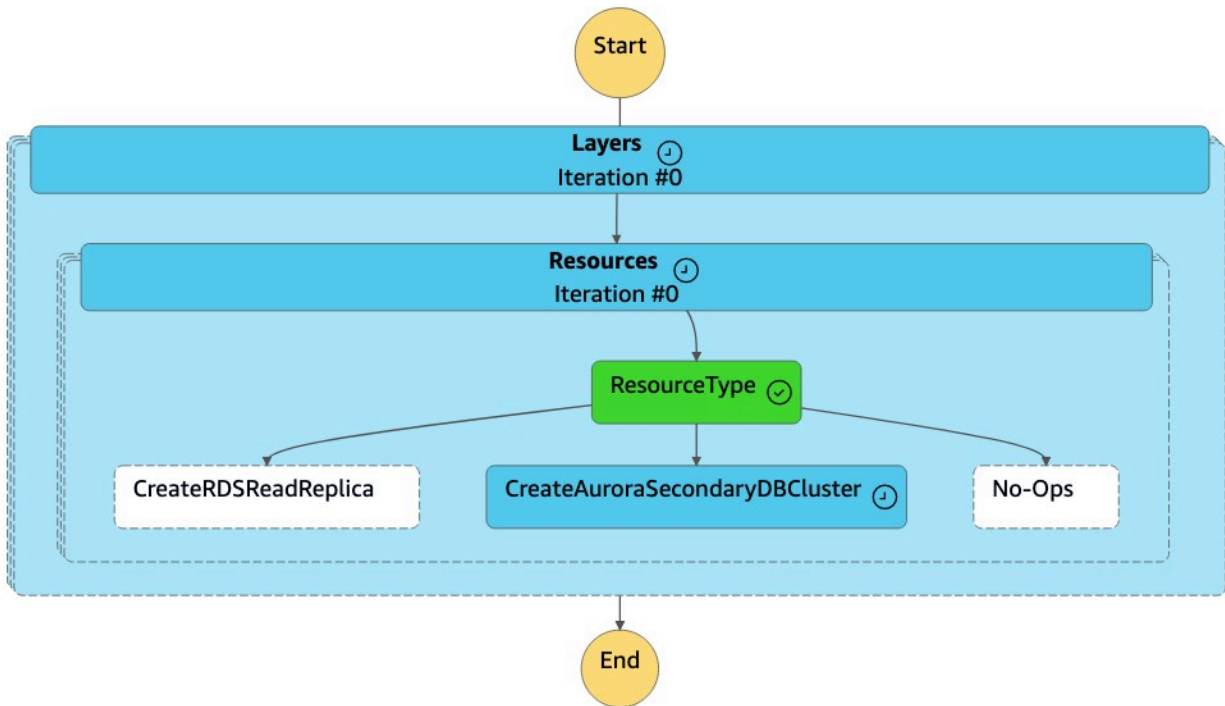
```
{
  "StatePayload": [
    {
      "layer": 1,
      "resources": [
        {
          "resourceType": "CreateAuroraSecondaryDBCluster",
          "resourceName": "To create secondary Aurora MySQL Global Database Cluster",
          "parameters": {
            "GlobalClusterIdentifier": "!Import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

```

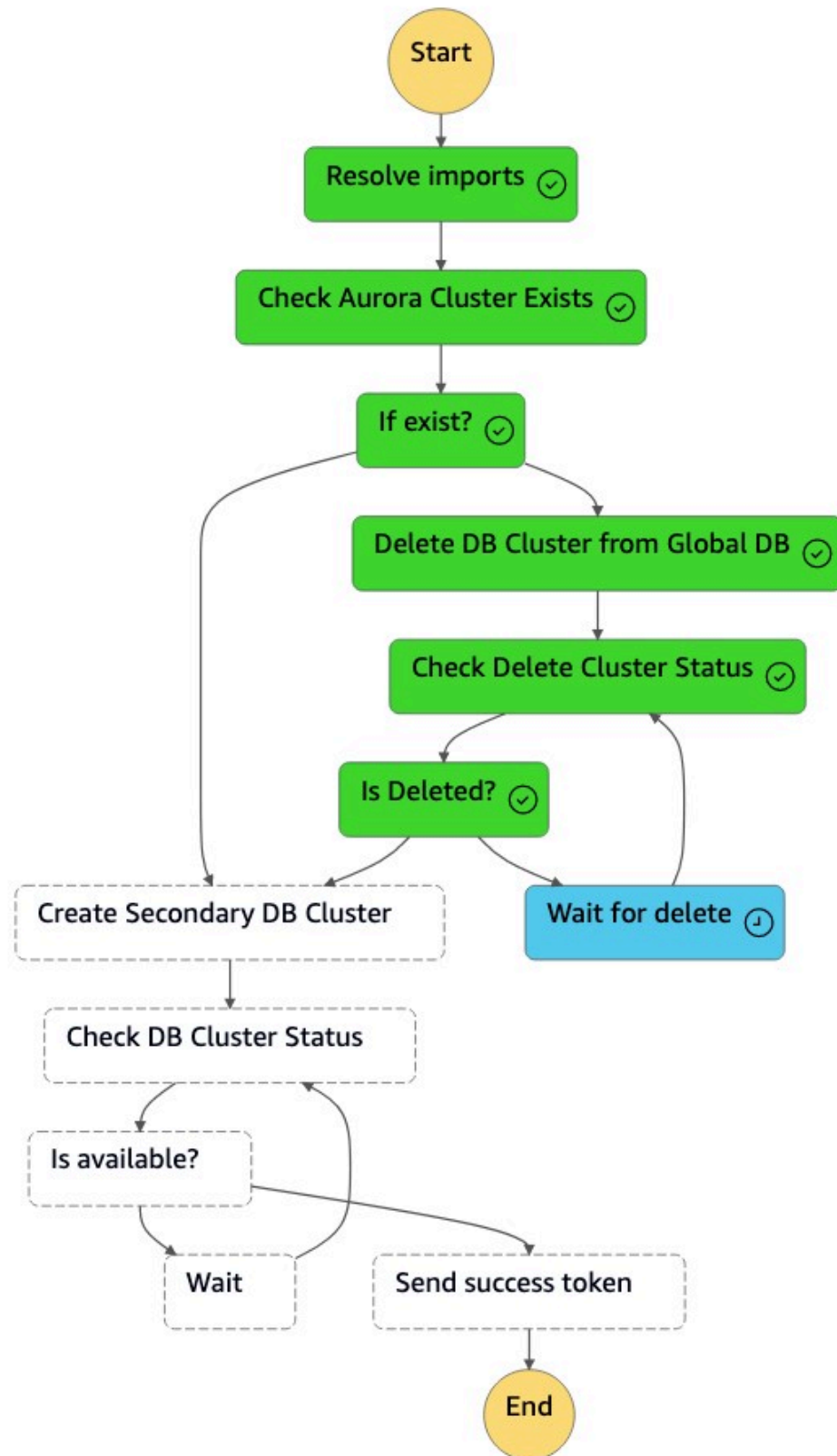
        "DBSubnetGroupName": "!Import dr-globaldb-cluster-mysql-subnet-group-
name",
        "VpcSecurityGroupIds": "!Import dr-globaldb-cluster-mysql-vpc-security-
group-ids",
        "Engine": "!Import dr-globaldb-cluster-mysql-engine",
        "EngineVersion": "!Import dr-globaldb-cluster-mysql-engine-version",
        "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"
    }
}
]
}
]
}

```

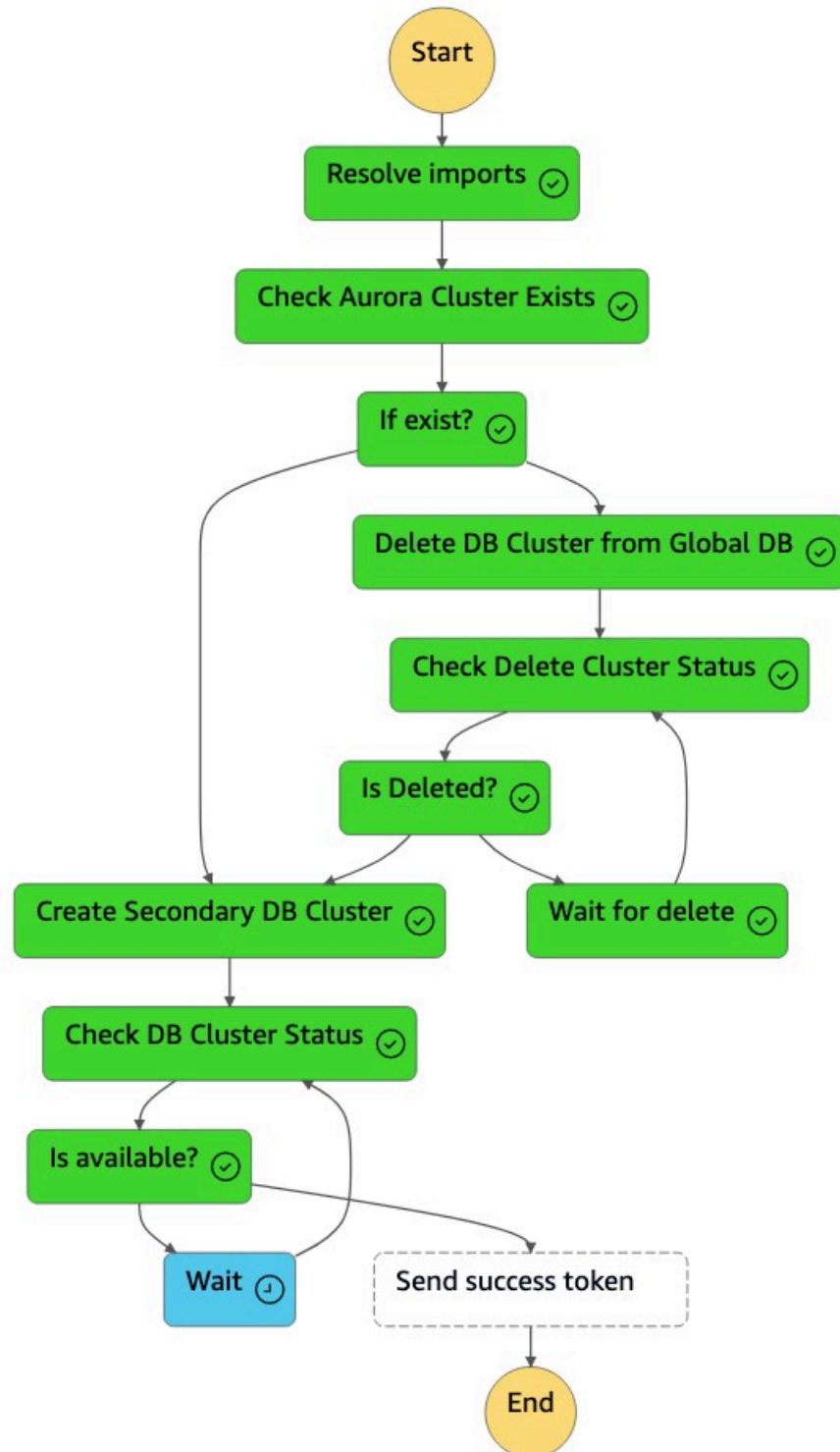
6. DR Orchestrator FAILBACK ステートマシンはリソースタイプを `!CreateAuroraSecondaryDBCluster`、`dr-orchestrator-stepfunction-create-Aurora-Secondary-cluster` ステートマシンを呼び出します。



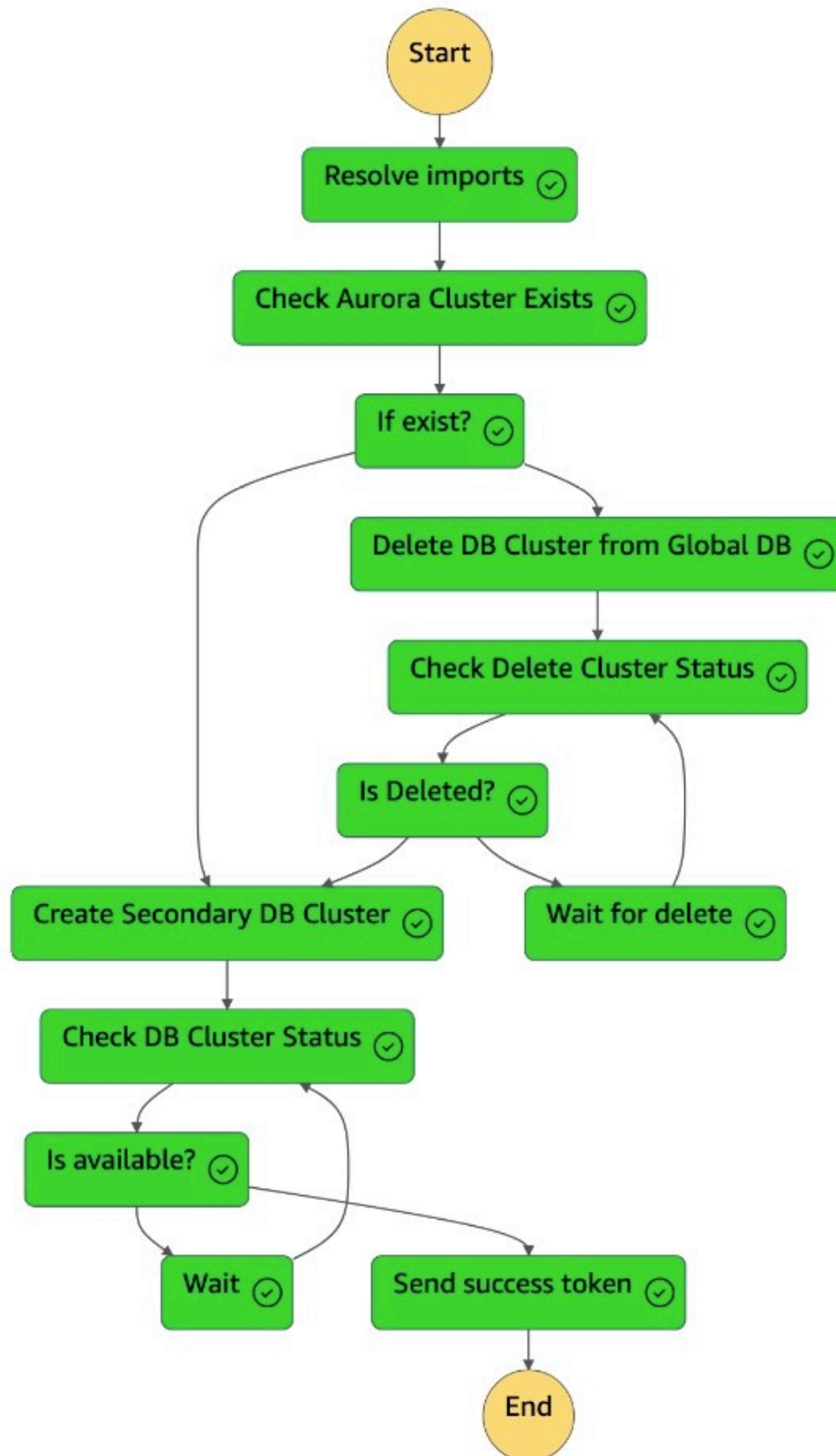
7. `dr-orchestrator-stepfunction-create-Aurora-Secondary-cluster` ステートマシンは、以前のプライマリリージョン (`dbcluster-01`) から既存のクラスター () を削除します `us-east-1`。



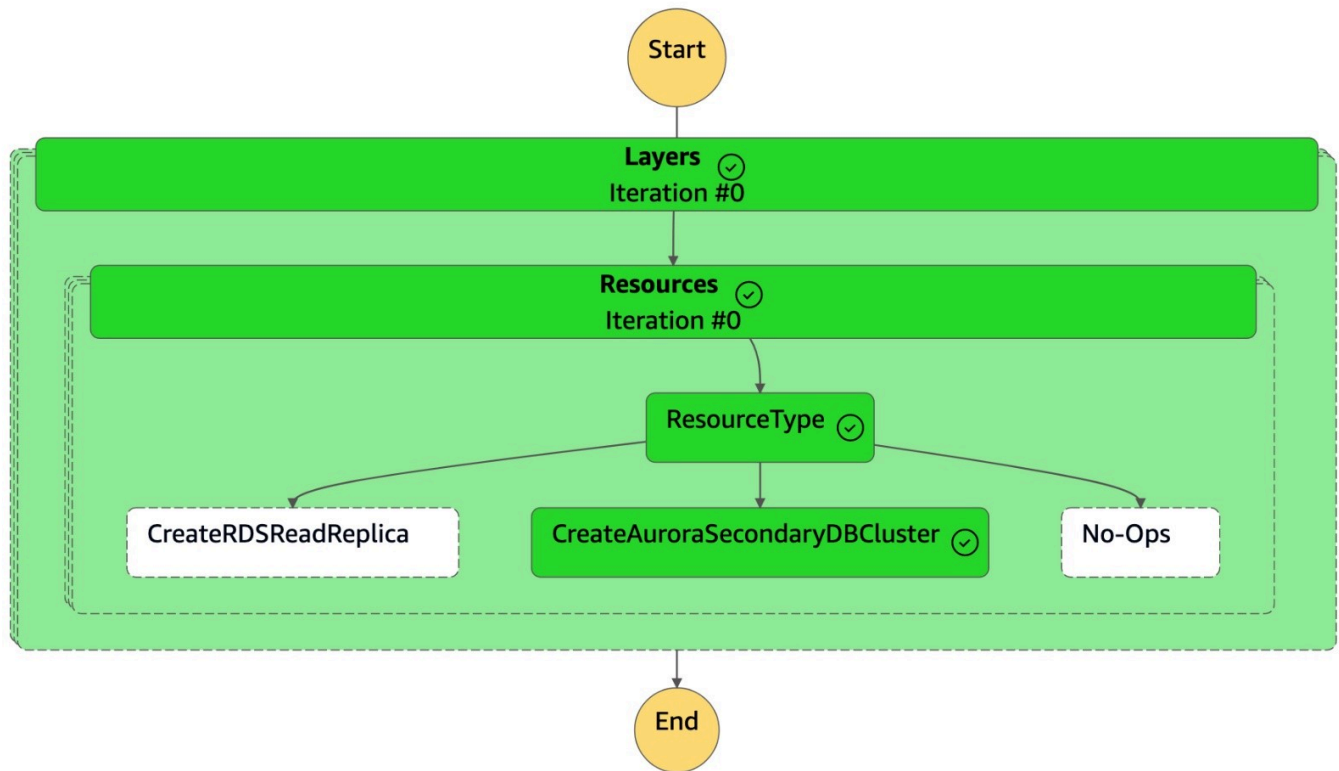
8. クラスター (dbcluster-01) が削除されると、ステートマシンは DB インスタンスとともに新しいクラスター (dbcluster-01) を作成し、Aurora グローバルデータベースをセカンダリクラスターとして結合して読み取り専用ワークロードを処理します。



9. セカンダリクラスターが利用可能になると、`dr-orchestrator-stepfunction-create-Aurora-Secondary-cluster` ステートマシンは完了し、成功トークンを DR Orchestrator Failback ステートマシンに送り返します。



10dr-orchestrator-stepfunction-FAILBACK ステートマシンが完了しました。



11Aurora グローバルデータベースは、Amazon RDS コンソールで確認できます。

プライマリ DB クラスターを us-east-1 に再配置する場合は、[「スイッチオーバー」](#) セクションで説明されている手順に従います。

ソリューションをデプロイする

前提条件を設定したら、[DR Orchestrator Framework パターン](#) の指示に従います。Amazon、RDS Aurora、および Amazon の手順が含まれています ElastiCache。ソリューションをテストするには、前提条件と制限を確認し、サンプルデータベースを にデプロイします AWS。

前提条件

- このソリューションをデプロイ [AWS アカウント](#) するアクティブな。
- [Amazon Simple Storage Service \(Amazon S3\)](#) バケットと [AWS CloudFormation](#) テンプレートで使われるリソースを作成するアクセス許可。
- DR Orchestrator FAILBACK ステートマシンを実行する前に：
 - [RDS インスタンスを手動で変更](#) するか、[Amazon Aurora DB クラスター を変更](#) して、 を無効にします DeletionProtection。
 - 根本原因分析 () を実行する予定がある場合は、Amazon RDS インスタンスまたは Aurora クラスターを削除する前に [DB スナップショット](#) を作成します RCA。

制約事項

- 機能の可用性とサポートは、各データベースエンジンの特定のバージョンと によって異なります AWS リージョン。クロスリージョンレプリケーションの機能とリージョンの可用性の詳細については、「[クロスリージョンリードレプリカ](#)」を参照してください。
- このソリューションは、1 つのリードレプリカを持つプライマリデータベースインスタンスを使用してテストされています。複数のリードレプリカを使用する場合は、本番環境に実装する前に、ソリューションを徹底的にテストしてください。
- Amazon の場合 RDS、このソリューションは次のエンジンのみをサポートします: [Amazon RDS for My SQL](#)、[Amazon RDS for Postgre SQL](#)、および [Amazon RDS for MariaDB](#)。

にサンプルデータベースをデプロイする AWS

上のデータベースのクロスリージョン DR ソリューションをデモンストレーションするには AWS、[Amazon RDS for My SQL](#)、[Amazon Aurora My SQL-Compatible Edition](#)、および [Amazon ElastiCache \(Redis OSS\) - Global Datastore](#) をデプロイできます。 [aws-cross-region-dr-databases](#) GitHub リポジトリの App-Stack README ファイルにある手順を使用します。この例では、プライマリリージョンは us-east-1、セカンダリリージョンは us-west-2。

アプリケーションスタックは、AWS CloudFormation [出力](#)セクションを使用して、DR Orchestrator Framework が AWS データベースのフェイルオーバーとフェイルバックを自動化するために使用するパラメータをエクスポートします。

よくある質問

フェイルオーバーとフェイルバックのための DR Orchestrator Framework に関する質問への回答を見つけます。

このアプローチを使用して達成できる RPO と RTO は何ですか？

目標復旧時間 (RTO) と目標復旧時点 (RPO) の詳細については、「データベースのディザスタリカバリ [AWS 戦略](#)」を参照してください。

AWS CloudFormation エクスポート変数の使用は必須ですか？

いいえ。Amazon Aurora グローバルデータベースまたは Amazon RDS DB インスタンスの値を JSON 形式で直接渡すことができます (例: - "RDSInstanceIdentifier": "rds-mysql-instance")。

DR オーケストレーターの FAILOVER ワークフローを使用して、複数の AWS データベースをフェイルオーバーできますか？

はい。入力パラメータファイルで複数のリソースを渡して、複数の AWS データベースをフェイルオーバーできます。次のコード例は、Amazon RDS for MySQL リードレプリカと Amazon ElastiCache (Redis OSS) グローバルデータストアを並行してフェイルオーバーしていることを示しています。

```
{
  "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"
  }
}
```

Amazon RDS の DR オーケストレーター FAILBACK ステートマシンを実行するときに InvalidParameterCombination エラーを回避するにはどうすればよいですか？

エラーの全文は次のとおりです。

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

エラーを回避するには、DR Orchestrator FAILBACKステートマシンを実行するDeleteProtection 前に を無効にして [RDS インスタンスを変更します](#)。

次のステップ

DR Orchestrator Framework を使用して、フェイルオーバーまたはフェイルバックアクティビティを開始する前に承認プロセスを追加できます。例えば、DR イベント中にフェイルオーバーを開始するときに、 の上にステートマシンを追加して、[Amazon Simple Notification Service \(Amazon SNS\)](#) 通知を E メールで DR Orchestrator Failover 送信できます。承認が付与されるとすぐに、フェイルオーバーアクティビティが開始されます。

アプリケーションのフェイルオーバーを DR Orchestrator Framework と統合して、DNS エントリを切り替え、アプリケーションをデータベースインスタンスまたはクラスターの新しいエンドポイントにポイントすることもできます。

[AWS X-Ray](#) トレースを使用して、RTO を計算するフェイルオーバー期間を取得できます。X-Ray 上にモニタリングダッシュボードを構築できます。

リソース

- [グローバルデータストア AWS リージョン を使用した 間のレプリケーション](#) (Amazon ElastiCache (Redis OSS) ドキュメント)
- [Amazon Aurora Global Database のフェイルオーバーを使用する](#) (Aurora ドキュメント)
- [計画的フェイルオーバーと計画外のフェイルオーバーの Amazon Aurora Global Database エンドポイント管理を自動化する](#) (ブログ記事)
- [ディザスタリカバリ: Terraform を使用した Amazon Aurora グローバルデータベースの 3 つのフェイルオーバーシナリオ \(パート 2\)](#) (ブログ記事)
- [のデータベースのディザスタリカバリ戦略 AWS](#) (AWS 規範ガイド戦略)
- [DR Orchestrator Framework を使用してクロスリージョンフェイルオーバーとフェイルバックを自動化する](#) (AWS 規範ガイドパターン)

ドキュメント履歴

以下の表は、本ガイドの重要な変更点について説明したものです。今後の更新に関する通知を受け取る場合は、[RSS フィード](#) をサブスクライブできます。

変更	説明	日付
初版発行	—	2024 年 5 月 3 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。