



Automatisieren Sie Ihre DR-Lösung für relationale Datenbanken auf AWS

AWS Prescriptive Guidance



AWS Prescriptive Guidance: Automatisieren Sie Ihre DR-Lösung für relationale Datenbanken auf AWS

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

Die Handelsmarken und die Handelsaufmachung von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, durch die Kunden irregeführt werden könnten oder Amazon in schlechtem Licht dargestellt oder diskreditiert werden könnte. Alle anderen Handelsmarken, die nicht Eigentum von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise zu Amazon gehören oder nicht, mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

Einführung	1
Ziele	2
-Übersicht	3
Der Failover-Prozess	4
DR Orchestrator-FAILOVER-Architektur	4
Der Failback-Prozess	5
DR Orchestrator-Failback-Architektur	5
DR Orchestrator-FAILOVER	7
Einzelheiten zu den Parametern	8
DR Orchestrator-FAILBACK	9
Aurora-Zustände und Step Functions Functions-Zustandsmaschinen	12
Stetiger Zustand	12
Status des Ereignisses	12
Umstellung	13
Manuelles ungeplantes Failover	18
Failback	22
Stellen Sie die Lösung bereit	31
Häufig gestellte Fragen	33
Welche RPO- und RTO-Werte kann ich mit diesem Ansatz erreichen?	33
Ist die Verwendung von AWS CloudFormation Exportvariablen obligatorisch?	33
Kann ich den DR Orchestrator FAILOVER-Workflow verwenden, um ein Failover für mehr als eine AWS Datenbank durchzuführen?	33
Wie kann ich den InvalidParameterCombination Fehler vermeiden, wenn ich die DR Orchestrator FAILBACK State Machine für Amazon RDS ausführe?	34
Nächste Schritte	35
Ressourcen	36
Dokumentverlauf	37
.....	xxxviii

Automatisieren Sie Ihre DR-Lösung für relationale Datenbanken auf AWS

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

Mai 2024 ([Geschichte der Dokumente](#))

Bei der Planung einer [Disaster Recovery \(DR\) -Strategie](#) für Amazon Web Services (AWS) können Sie eine DR-Lösung für die Datenbanken in Ihrem Unternehmen implementieren. Sie können die DR-Lösung automatisieren, indem Sie eine ereignisgesteuerte Architektur für den Failover der Datenbank-Instances konfigurieren. Zu den Vorteilen dieses Ansatzes gehören:

- Reduzierung menschlicher Fehler, die durch manuelle Koordination verursacht werden
- In einigen Fällen wird das kürzest mögliche Wiederherstellungszeitziel (Recovery Time Objective, RTO) erreicht
- Erleichterung wiederholbarer Tests Ihrer DR-Lösung

Ohne Automatisierung würde ein Ausfall in Ihrer Primäranlage Ressourcen in Ihrem Unternehmen erfordern AWS-Region, um Schritte in einem Runbook manuell durchzuführen. Diese Schritte können mehrere Prozesse umfassen, z. B. das Heraufstufen von Amazon Relational Database Service (Amazon RDS) -Read Replicas in der Sekundärdatenbank AWS-Region , sodass sie als neue primäre Datenbank-Instance dienen. AWS bietet [AWS SDK for Python \(Boto3\)](#) APIs zur Automatisierung dieser Aktionen, die die Bausteine einer DR-Lösung für die Bedürfnisse Ihres Unternehmens bilden können.

In diesem Leitfaden wird ein zentralisierter, manifestgesteuerter DR-Orchestrator beschrieben, der Ihnen helfen kann, eine DR-Lösung für AWS Datenbanken innerhalb derselben Datenbank zu entwickeln. AWS-Regionen AWS-Konto Die Entscheidung, einen Failover durchzuführen, würde immer noch menschliches Eingreifen erfordern und ist nicht automatisierbar.

Die Zielgruppen dieses Leitfadens sind Anwendungsarchitekten, Infrastrukturarchitekten, Datenbankarchitekten und Datenbankberater.

Ziele

Durch die Implementierung der DR Orchestrator Framework-Lösung können Sie die folgenden Ergebnisse erzielen:

- Reduzieren Sie manuelle Fehler bei einem Ausfall — Die manuelle Ausführung von DR-Prozessen kann fehleranfällig sein. Die Automatisierung von Aktionen während der Phasen einer DR-Aktivität reduziert die Anzahl manueller Fehler und trägt zu einem reibungsloseren Failover oder Failback bei.
- DR in großem Umfang verwalten — Wenn Ihr Unternehmen über eine große Flotte von Datenbanken verfügt, hilft Ihnen Building Automation for DR dabei, Ihre DR-Lösung zu skalieren. Sie können Ihre AWS Datenbank-DR-Lösung mithilfe des DR Orchestrator Framework automatisieren. DR Orchestrator automatisiert die Schritte für Failover oder Failback von AWS Datenbanken mithilfe einer Parameterdatei im JSON-Format.

Überblick über das DR Orchestrator Framework

DR Orchestrator Framework bietet eine Ein-Klick-Lösung zur Orchestrierung und Automatisierung regionsübergreifender DR für Datenbanken. AWS Es verwendet [AWS Step Functions](#) und [AWS Lambda](#) führt die erforderlichen Schritte während des Failovers und des Failbacks aus. Die Step Functions Zustandsmaschinen bilden die Grundlage für die Entscheidungsfindung innerhalb des Orchestrator-Designs. Die API-Operationen zur Durchführung von Failover- oder Failback-Aktionen sind in Lambda-Funktionen codiert, die von der Zustandsmaschine aus aufgerufen werden. Die Lambda-Funktionen führen [AWS SDK für Python \(Boto3\)](#) APIs aus, um mit AWS Datenbanken zu interagieren.

Das DR Orchestrator Framework enthält zwei Hauptzustandsmaschinen, die den Failover- und Failback-Phasen entsprechen.

Bei Amazon RDS wird in der Failover-Phase eine regionsübergreifende RDS-Read Replica in eine eigenständige DB-Instance umgewandelt. Bei Amazon Aurora ist der Writer-Knoten nicht verfügbar, wenn die primäre Region während eines seltenen, unerwarteten Ausfalls ausgefallen ist. Die Replikation zwischen dem Writer-Knoten und den sekundären Clustern wird gestoppt. Sie müssen den sekundären Cluster von der globalen Datenbank trennen und ihn zu einem eigenständigen Cluster heraufstufen. Anwendungen können eine Verbindung herstellen und Schreibdatenverkehr an den eigenständigen Cluster senden. Sie können denselben Prozess verwenden, um vom primären DB-Cluster der globalen Datenbank zu den sekundären Regionen zu [wechseln](#). Verwenden Sie diesen Ansatz für kontrollierte Szenarien wie die folgenden:

- Betriebliche Wartung
- Geplante Betriebsabläufe
- Förderung eines sekundären Clusters von Amazon ElastiCache (Redis OSS) als Ihrem neuen primären Cluster

In der Failback-Phase wird die Live-Replikation von Daten zwischen einer primären Live-Region und einer neuen sekundären Region eingerichtet.

Es ist wichtig zu verstehen, dass DR Orchestrator nur für Datenbanken gilt. Alle Anwendungen, die auf diese Datenbanken verweisen und sich in derselben Region befinden, benötigen möglicherweise eine separate Tandem-Failover-Lösung. Nach dem Failover der Datenbanken in die sekundäre

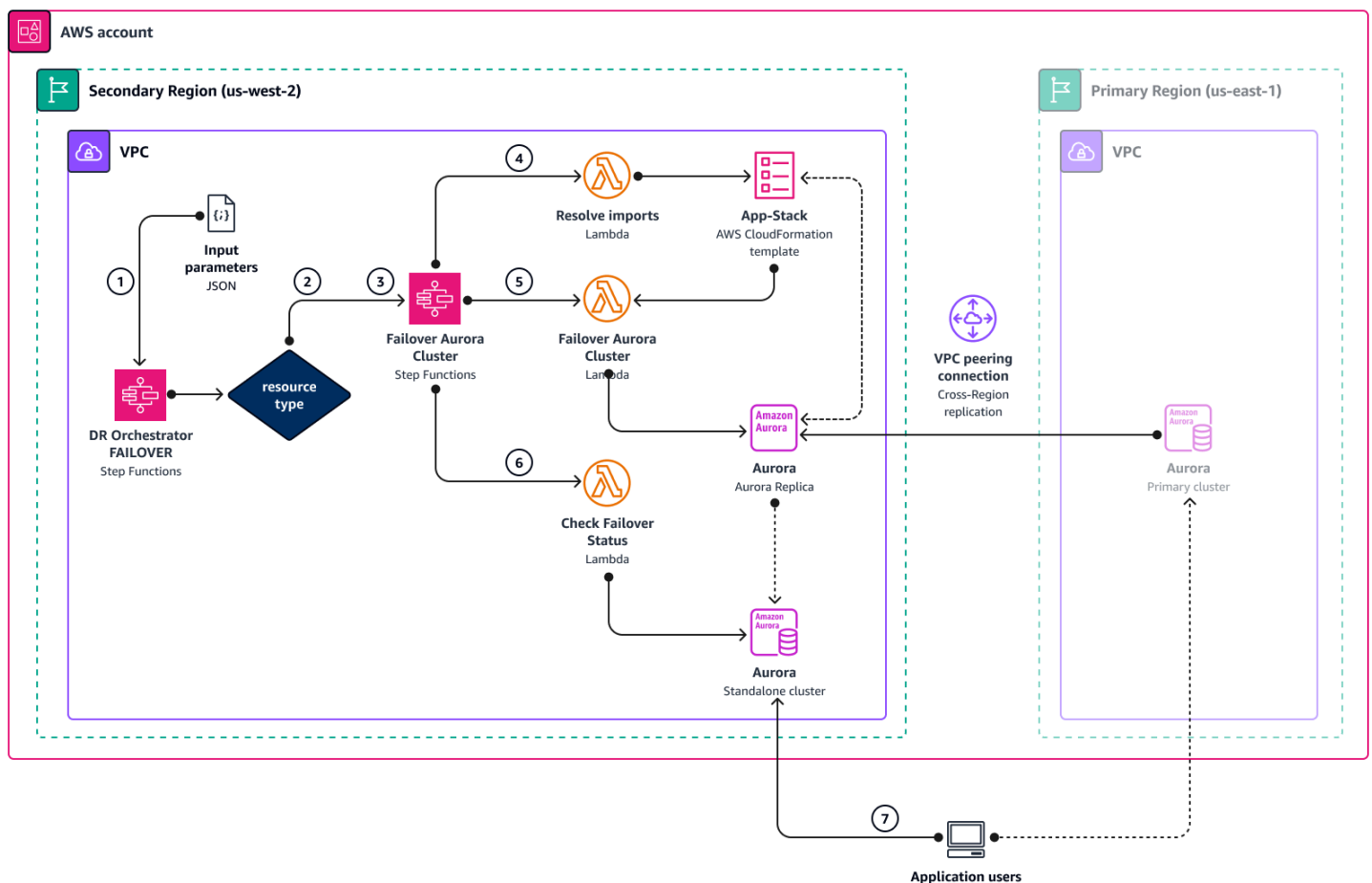
Region müssen die Anwendungen aktualisiert werden, um eine Verbindung zu den neuen Datenbankinstanzen herzustellen, die als Datenquelle dienen.

Der Failover-Prozess

Führen Sie die DR Orchestrator FAILOVER Zustandsmaschine aus, um einen Failover durchzuführen. Zu diesem Zeitpunkt ist in der sekundären Region bereits eine sekundäre Datenbank vorhanden, entweder als Read Replica (Amazon RDS) oder als sekundärer Cluster (Amazon Aurora). Wenn Sie die DR Orchestrator FAILOVER Zustandsmaschine ausführen, wird die sekundäre Datenbank zur primären Datenbank heraufgestuft.

DR Orchestrator-FAILOVER-Architektur

Das folgende Diagramm zeigt die Konzepte des Failover-Prozesses für Amazon Aurora bei Verwendung von DR Orchestrator. Amazon Aurora und Amazon ElastiCache verwenden denselben Workflow, jedoch mit unterschiedlichen Zustandsmaschinen und Lambda-Funktionen.



1. Die `DR Orchestrator FAILOVER` Zustandsmaschine liest die JSON-Eingabeparameter.
2. Basierend auf dem `resourceType` Parameter ruft die Zustandsmaschine andere Zustandsmaschinen auf: `Promote RDS Read ReplicaFailover Aurora Cluster`, `oderFailover ElastiCache`. Wenn in der Eingabe mehr als eine Ressource übergeben wird, laufen diese Zustandsmaschinen parallel.
3. Die `Failover Aurora Cluster` Zustandsmaschine ruft Lambda-Funktionen in jedem der folgenden drei Schritte auf.
4. Die `Resolve imports` Lambda-Funktion wird `"! import <export-variable-name>"` mit den tatsächlichen Werten aus der App-Stack AWS CloudFormation Vorlage aufgelöst.
5. Die **Failover Aurora Cluster** Lambda-Funktion bewirbt die Read Replica als eigenständige DB-Instance.
6. Die **Check Failover Status** Lambda-Funktion überprüft den Status der beworbenen DB-Instance. Sobald der Status VERFÜGBAR ist, sendet die Lambda-Funktion ein Erfolgstoken zurück an die aufrufende Zustandsmaschine und schließt den Vorgang ab.
7. Sie können Ihre Anwendungen zur eigenständigen Datenbank in der DR-Region (us-west-2) umleiten, die jetzt die primäre Datenbank ist.

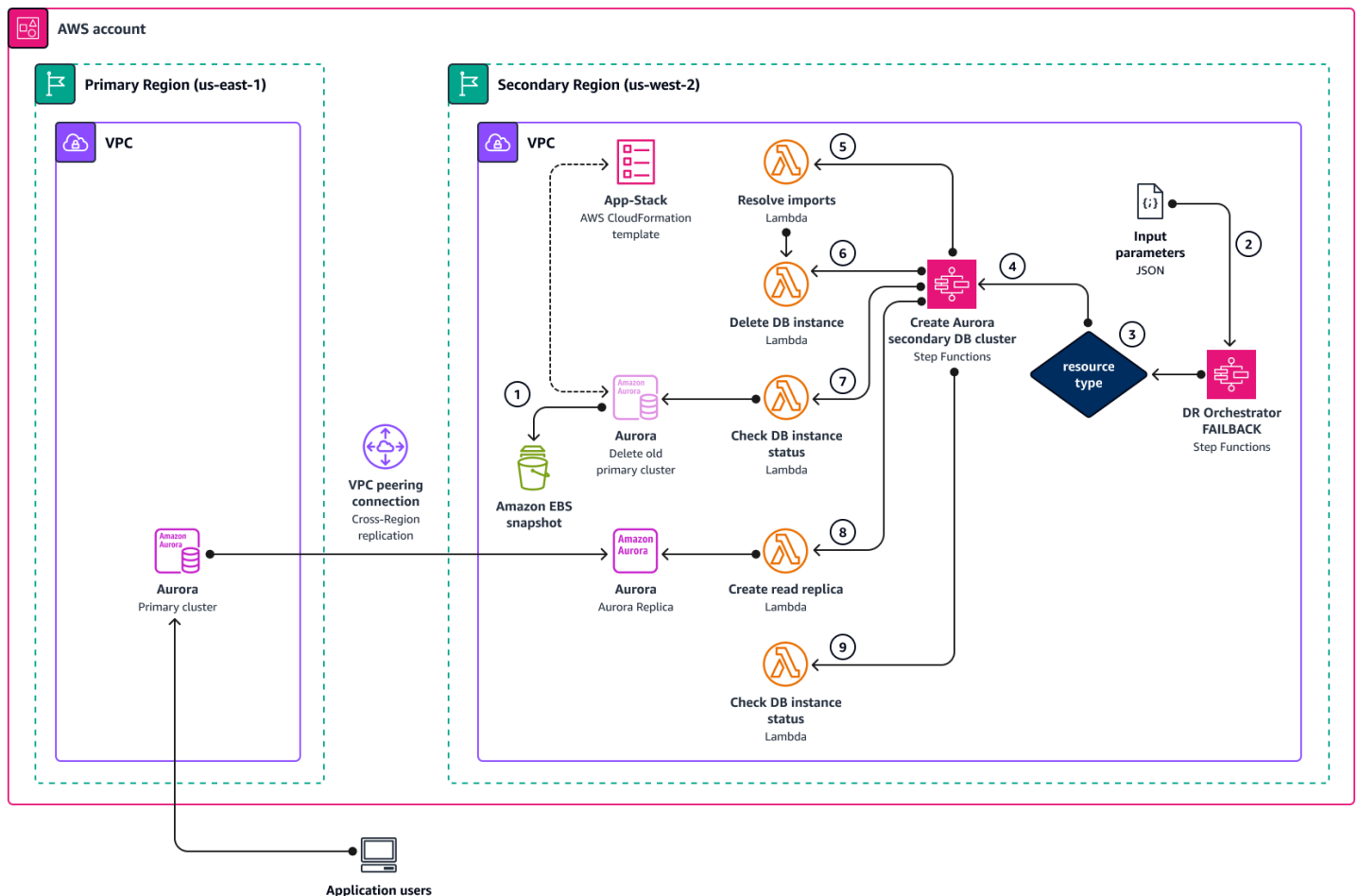
Der Failback-Prozess

Nachdem Ihre frühere primäre Region (us-east-1) wieder aktiv ist, können Sie zu ihr zurückkehren, sodass die Datenbank in wieder zur primären Region us-east-1 wird. Um das Failback zu starten, führen Sie die `DR Orchestrator FAILBACK` Zustandsmaschine aus. Wie der Name schon sagt, beginnt dieser Zustandsmaschine damit, Änderungen in Ihrer neuen primären Region (us-west-2) zurück in die frühere primäre Region (us-east-1) zu replizieren, die als aktuelle sekundäre Region fungiert.

Nachdem die Replikation zwischen den beiden Regionen eingerichtet wurde, können Sie das Failback einleiten. Um ein Failback durchzuführen und zu Ihrer ursprünglichen primären Region (us-east-1) zurückzukehren, führen Sie den `DR Orchestrator FAILOVER` Zustandsmaschine in der aktuellen sekundären Region (us-east-1) aus, um sie zur primären Region hochzustufen.

DR Orchestrator-Failback-Architektur

Das folgende Diagramm zeigt die Konzepte des Failback-Prozesses für Amazon Aurora bei Verwendung von DR Orchestrator.



1. Bevor Sie mit dem Failback beginnen, erstellen Sie einen manuellen DB-Snapshot, den Sie bei der Root Cause Analysis (RCA) verwenden können.

Deaktivieren Sie außerdem das DeletionProtection für den Aurora-Cluster in der vorherigen primären Region (us-east-1).

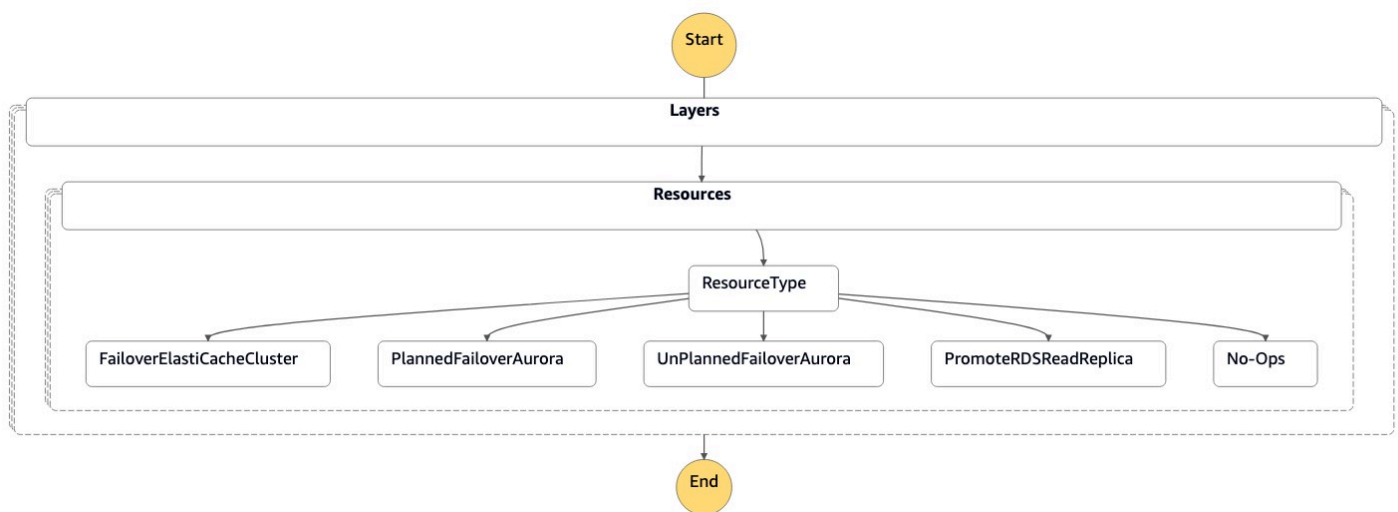
2. Die DR Orchestrator FAILBACK Zustandsmaschine liest die JSON-Eingabeparameter.
3. Basierend auf dem resourceType ruft die DR Orchestrator FAILBACK Zustandsmaschine die Create Aurora Secondary DB Cluster Zustandsmaschine auf.
4. Die Create Aurora Secondary DB Cluster Zustandsmaschine ruft Lambda-Funktionen in jedem der folgenden fünf Schritte auf.
5. Die Resolve import Lambda-Funktion wird "! import <export-variable-name>" mit den tatsächlichen Werten aus der App-Stack CloudFormation Vorlage aufgelöst.
6. Die Delete DB Instance Lambda-Funktion löscht die frühere Primärinstanz.

7. Die Check DB instance status Lambda-Funktion überprüft die, Delete DB Instance status bis die Datenbank gelöscht wird.
8. Die Create Read Replica Lambda-Funktion erstellt eine Read Replica in der sekundären Region aus der DB-Instance, die sich in der neuen primären Region befindet.
9. Die Check DB instance status Lambda-Funktion überprüft den Status der Read Replica-DB-Instance. Wenn der Status VERFÜGBAR ist, sendet die Lambda-Funktion ein Erfolgstoken zurück an die aufrufende Zustandsmaschine, die abgeschlossen ist.

DR Orchestrator-FAILOVER

Verwenden Sie die DR Orchestrator FAILOVER Zustandsmaschine im DR-Ereignis, wenn die primäre Region (us-east-1) ausgefallen ist, oder bei geplanten Ereignissen wie Wartungsarbeiten.

Die Funktion kann aufgerufen werden, um ein Failover einzelner oder mehrerer Datenbanken parallel durchzuführen.



Die Zustandsmaschine akzeptiert Parameter im JSON-Format, wie im folgenden Code dargestellt:

```

{
  "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"
      }
    }
  ]
}
```

Einzelheiten zu den Parametern

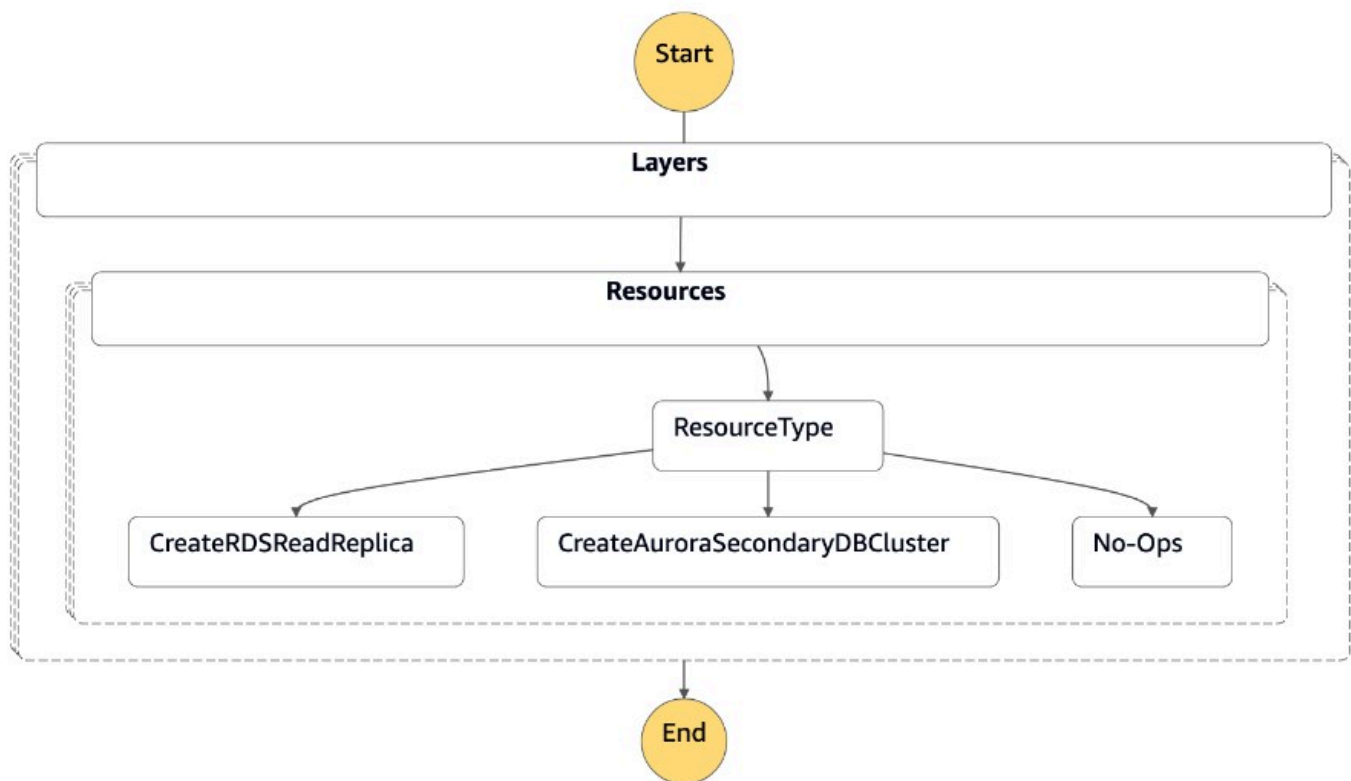
Die folgende Tabelle zeigt die von der DR Orchestrator FAILOVER Zustandsmaschine verwendeten Parameter.

Parametername	Description	Erwartete Werte
layer(erforderlich: Zahl)	Die Reihenfolge der Verarbeitung. Alle in Layer 1 definierten Ressourcen müssen ausgeführt werden, bevor die Layer-2-Ressourcen ausgeführt werden.	1 oder 2 usw.
Ressourcen (erforderlich: Wörterbuch-Array)	Alle Ressourcen innerhalb einer einzigen Schicht laufen parallel.	<pre>{ "resourceType": " String", "resourceName": "St ring",</pre>

Parametername	Description	Erwartete Werte
		<pre>"parameters":{ "<param1>":"<!Import cft-output-1">, }</pre>
resourceType (erforderlich: Zeichenfolge)	Typ der Ressource zur Identifizierung der Ressource	PromoteRDSReadReplica oder FailoverElastiCacheCluster
resourceName (optional: Zeichenfolge)	Um zu ermitteln, zu welchem Anwendungsportfolio diese Ressourcen gehören	Promote RDS for MySQL Read Replica
Parameter (erforderlich: Wörterbuch-Array)	Liste der Parameter, die für ein Failover oder ein Failback der AWS Datenbank erforderlich sind	<pre>{ "<param1>": "<!Import cft-output-1">, "<param2>": "<!Import cft-output-2">, }</pre>

DR Orchestrator-FAILBACK

Verwenden Sie die DR Orchestrator FAILBACK Zustandsmaschine nach dem DR-Ereignis, wenn die frühere primäre Region (us-east-1) aktiv ist. Sie können die [Read Replica](#) für Amazon RDS in der früheren primären Region aus der neuen primären Region (us-west-2) erstellen, um Ihrer DR-Strategie zu entsprechen. Da es sich um eine geplante Veranstaltung handelt, können Sie diese Aktivität auf das Wochenende oder außerhalb der Hauptverkehrszeiten mit voraussichtlicher Ausfallzeit planen.



Die Zustandsmaschine akzeptiert Parameter im JSON-Format, wie im folgenden Code dargestellt:

```
{
  "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-Zustände und Step Functions Functions-Zustandsmaschinen

In diesem Abschnitt werden die Prozess- und Zustandsmaschinen behandelt, die für Failover und Failback von Amazon Aurora Aurora-Clustern spezifisch sind. Die Cluster sind als globale Datenbank konfiguriert.

Note

Zu Demonstrationszwecken wird in diesem Beispiel die Aurora MySQL-Compatible Edition verwendet. Sie können ähnliche Schritte für die Aurora PostgreSQL-Compatible Edition verwenden.

Stetiger Zustand

Im Steady-State wurde eine MySQL-Compatible globale Amazon Aurora Aurora-Datenbank (`dr-globaldb-cluster-mysql`) mit zwei DB-Clustern erstellt. Der erste DB-Cluster (`db-cluster-01`) wurde im primären AWS-Region (`us-east-1`) erstellt, um die read/write Arbeitslast zu bedienen. Der zweite DB-Cluster (`db-cluster-02`) wurde in der sekundären Region (`us-west-2`) erstellt, um den schreibgeschützten Workload zu verwalten.

Zusätzlich zur Bereitstellung der DR-Lösung können Sie die Belastung Ihres primären DB-Clusters reduzieren, indem Sie Leseanfragen von Ihren Anwendungen an den sekundären DB-Cluster weiterleiten. Jeder dieser Cluster enthält eine Datenbank-Instance namens `dbcluster-01-use1-instance-1` und `dbcluster-02-usw2-instance-2`.

Status des Ereignisses

Durch die Verwendung einer globalen Amazon Aurora Aurora-Datenbank können Sie Katastrophen relativ schnell planen und sich nach einer Katastrophe erholen. Die Wiederherstellung nach einem Notfall wird in der Regel anhand von Werten für Recovery Time Objective (RTO) und Recovery Point Objective (RPO) gemessen. Weitere Informationen finden Sie unter [Verwenden von Switchover oder Failover in einer globalen Amazon Aurora Aurora-Datenbank](#).

Bei einer globalen Aurora-Datenbank gibt es zwei verschiedene Ansätze für Failover:

- Switchover (verwaltetes geplantes Failover)
- Failover (manuelles, ungeplantes Failover oder Trennen und Heraufstufen)

Umstellung

Switchover ist für kontrollierte Umgebungen vorgesehen, z. B. für betriebliche Wartungsarbeiten und andere geplante Betriebsabläufe. Mithilfe eines verwalteten geplanten Failovers können Sie den primären DB-Cluster Ihrer globalen Aurora-Datenbank in eine der sekundären Regionen verlagern. Da Switchover wartet, bis die sekundären DB-Cluster mit der Primärdatenbank synchronisiert sind, ist RPO 0 (kein Datenverlust). Weitere Informationen finden Sie unter [Durchführen von Switchovers für globale Amazon Aurora Aurora-Datenbanken](#).

Die `dr-orchestrator-stepfunction-FAILOVER` Zustandsmaschine wird während des Ereignisstatus aufgerufen, um Ihren primären Cluster in die von Ihnen gewählte sekundäre Region umzuschalten (`us-west-2`).

Gehen Sie wie folgt vor, um den Switchover durchzuführen:

1. Melden Sie sich bei der AWS-Managementkonsole an.
2. Ändern Sie die Region in die DR-Region (`us-west-2`).
3. Navigieren Sie zu Services und wählen Sie Step Functions aus.
4. Navigieren Sie zur `dr-orchestrator-stepfunction-FAILOVER` Zustandsmaschine.
5. Wählen Sie Ausführung starten und geben Sie den folgenden JSON-Code in den Input - optional Abschnitt ein:

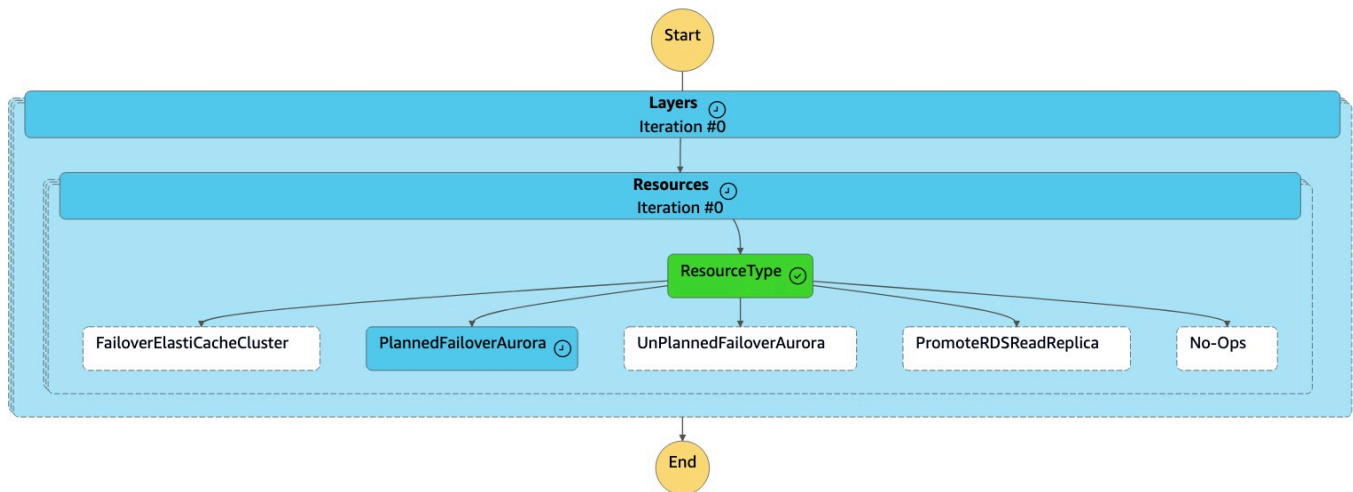
```
{
  "StatePayload": [
    {
      "layer": 1,
      "resources": [
        {
          "resourceType": "PlannedFailoverAurora",
          "resourceName": "Switchover (planned failover) of Amazon Aurora global
databases (MySQL)",
          "parameters": {
            "GlobalClusterIdentifier": "!Import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

```

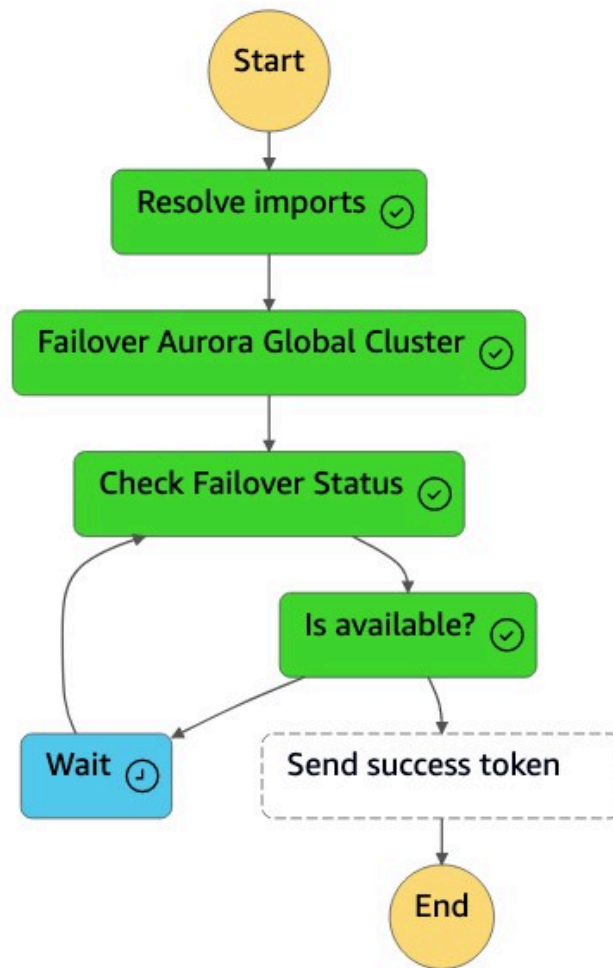
        "DBClusterIdentifier": "!Import dr-globaldb-cluster-mysql-cluster-
identifizier"
    }
}
]
}
]
}

```

6. Die `dr-orchestrator-stepfunction-FAILOVER` Zustandsmaschine liest den Ressourcentyp als `PlannedFailoverAuroraMySQL` und ruft die `dr-orchestrator-stepfunction-planned-Aurora-failover` Zustandsmaschine auf, um ein Failover für die globale Aurora-Datenbank durchzuführen.



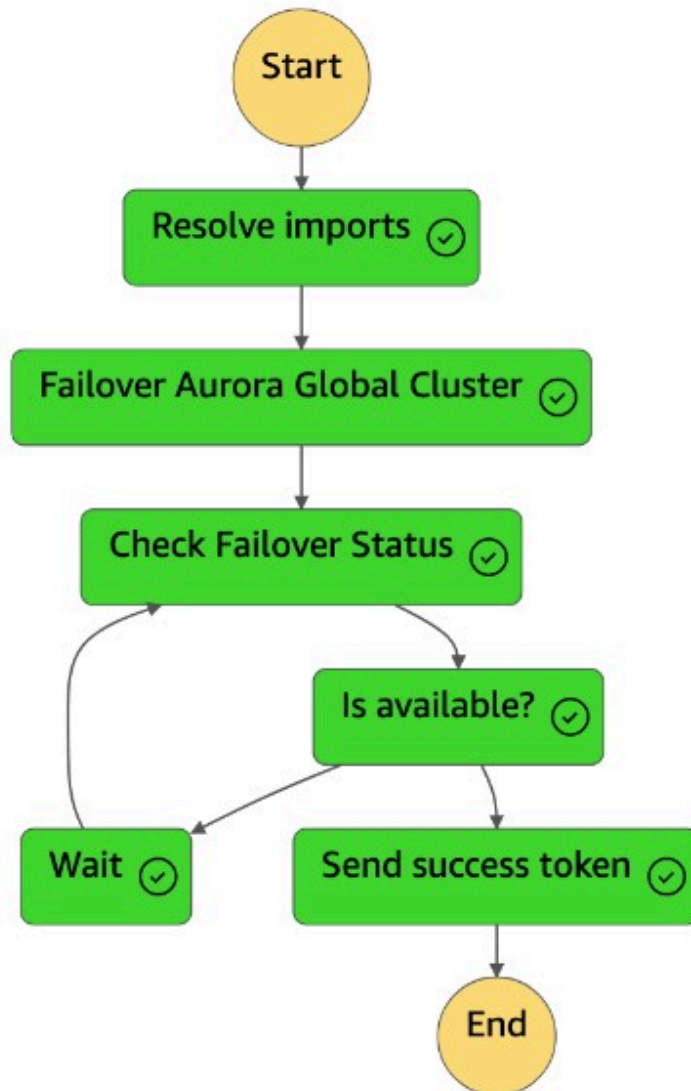
7. Die `dr-orchestrator-stepfunction-planned-Aurora-failover` Zustandsmaschine führt die folgenden Schritte aus, um die MySQL-Compatible globale Aurora-Datenbankrolle umzuschalten.



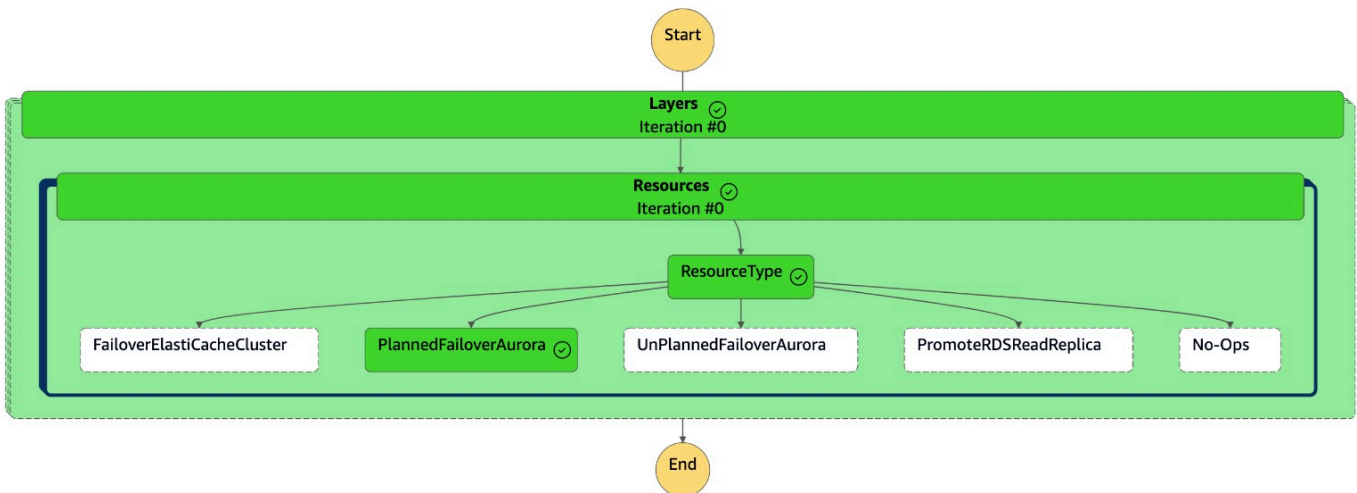
Schritt	Description	Erwartete Werte
Importe lösen	Eine Lambda-Funktion ersetzt !Import <variable name> Werte durch den tatsächlichen Namen.	<code>"!Import dr-global db-cluster-mysql-global-identifizier"</code> wird ersetzt durch <code>"dr-global db-cluster-mysql"</code> .

Schritt	Description	Erwartete Werte
Globaler Aurora-Failover-Cluster	Eine Lambda-Funktion ruft die Boto3-APIs failover_global_cluster auf, um ein Failover für die globale Aurora-Datenbank durchzuführen.	<pre>{ 'GlobalCluster': { 'GlobalClusterIdentifier': 'dr-global-cluster-mysql', 'GlobalClusterResourceId': 'cluster-cce7f9bec2846db4', 'GlobalClusterArn': 'arn:aws:rds::xxx', 'Status': 'failing-over', } }</pre>
Überprüfen Sie den Failover-Status	Eine Lambda-Funktion ruft describe_db_clusters Boto3-APIs auf, um den Status des Failovers zu überprüfen.	modifizieren, verfügbar
Erfolgstoken senden	Eine Lambda-Funktion ruft send_task_success Boto3-APIs auf und sendet ein Erfolgstoken zurück an die Zustandsmaschine. DR Orchestrator Failover	h7XRiCdLtd/83p1E0dMccoxIzFhglsdkzpK9mBVKZsp7d9yrT1W

8. Navigieren Sie zur Amazon RDS-Konsole. Unter Status ändern sich die Werte für die globale Aurora-Datenbank von Verfügbar auf Umschalten oder Ändern.
9. Nachdem die `dr-orchestrator-stepfunction-planned-Aurora-failover` Zustandsmaschine abgeschlossen ist, sendet sie ein Erfolgstoken zurück an die `dr-orchestrator-stepfunction-FAILOVER` Zustandsmaschine.



10 Die `dr-orchestrator-stepfunction-FAILOVER` Zustandsmaschine ist abgeschlossen.



Auf der Konsole hat der sekundäre Cluster (`dbc1uster-02`) jetzt die Rolle des primären Clusters, und der Cluster ist bereit, read/write Workloads zu verarbeiten. Die Rolle des ursprünglichen primären Clusters (`dbc1uster-01`) ist jetzt als Sekundärer Cluster aufgeführt.

Manuelles ungeplantes Failover

In seltenen Fällen kann es bei Ihrer globalen Aurora-Datenbank zu einem unerwarteten Ausfall der AWS-Region Primärdatenbank kommen. In einem solchen Fall sind der primäre Aurora-DB-Cluster und sein Writer-Knoten nicht verfügbar, und die Replikation zwischen dem primären Cluster und den sekundären Clustern wird eingestellt. Um sowohl Ausfallzeiten (RTO) als auch Datenverlust (RPO) zu minimieren, sollten Sie schnell ein regionsübergreifendes Failover durchführen und Ihre globale Aurora-Datenbank rekonstruieren. Weitere Informationen finden Sie unter [Wiederherstellung einer globalen Amazon Aurora Datenbank nach einem ungeplanten Ausfall](#).

Um einen ungeplanten Failover durchzuführen, müssen Sie Ihren sekundären Cluster von der globalen Aurora-Datenbank trennen. Bevor Sie den ungeplanten Failover durchführen, beenden Sie Anwendungsschreibvorgänge auf Ihrem primären Aurora-DB-Cluster. Nachdem der Failover erfolgreich abgeschlossen wurde, konfigurieren Sie die Anwendung neu, sodass sie in den neuen primären DB-Cluster schreibt. Dieser Ansatz trägt dazu bei, Datenverlust zu verhindern. Es hilft auch, Dateninkonsistenzen zu vermeiden, wenn der primäre Writer-Knoten während des Failover-Prozesses wieder online geht.

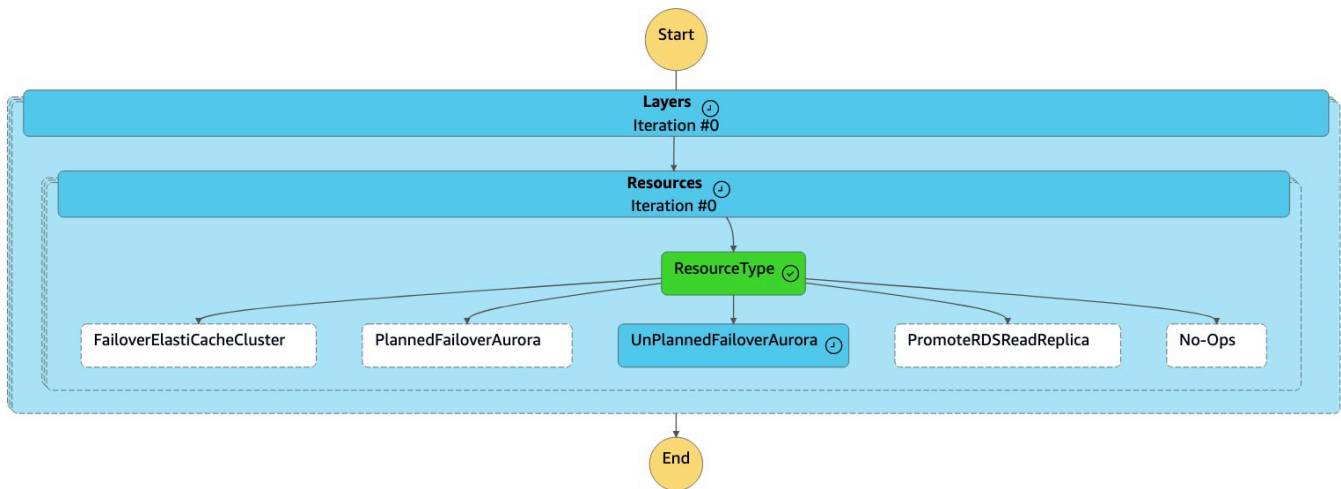
Rufen Sie den State Machine auf, um den ungeplanten Failover durchzuführen. `dr-orchestrator-stepfunction-FAILOVER` In diesem Beispiel befindet sich der sekundäre Cluster (`db-cluster-02`) in der DR-Region (`us-west-2`) im Steady-State.

Gehen Sie wie folgt vor, um den Failover durchzuführen:

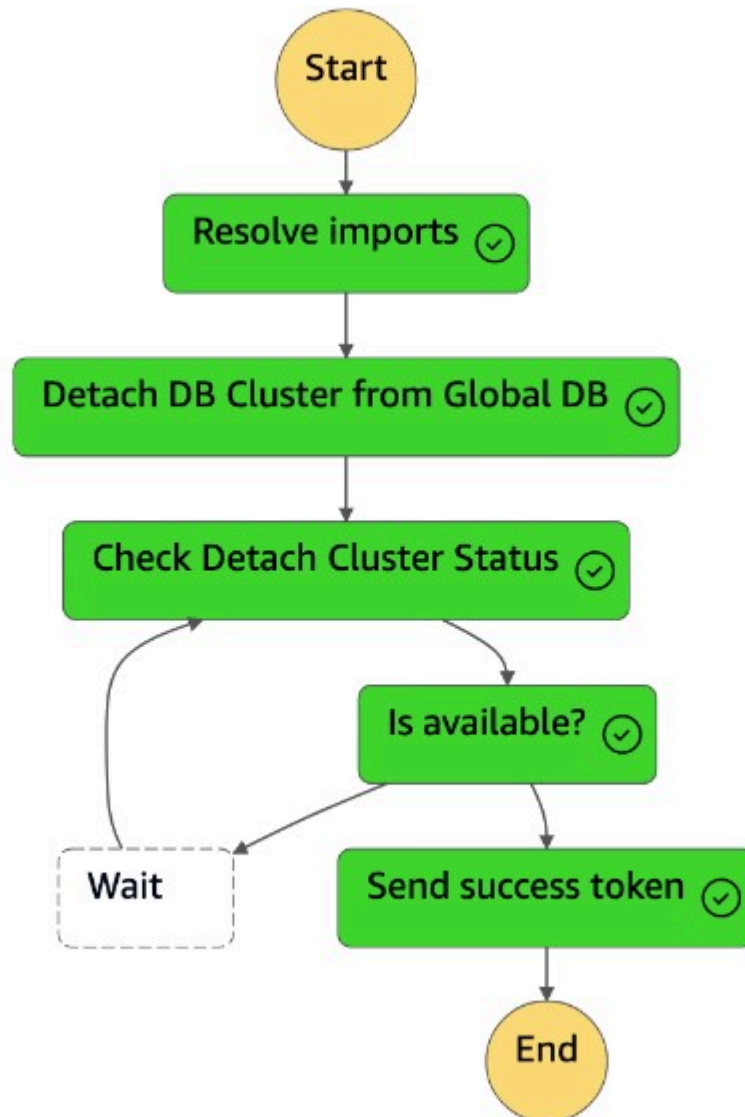
1. Melden Sie sich bei der -Konsole an.
2. Ändern Sie die Region in die DR-Region (us-west-2).
3. Navigieren Sie zu Services und wählen Sie Step Functions aus.
4. Navigieren Sie zur `dr-orchestrator-stepfunction-FAILOVER` Zustandsmaschine.
5. Wählen Sie Ausführung starten und geben Sie den folgenden JSON-Code in den Input - optional Abschnitt ein. Verwenden Sie `UnPlannedFailoverAurora` dabei `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"
          }
        }
      ]
    }
  ]
}
```

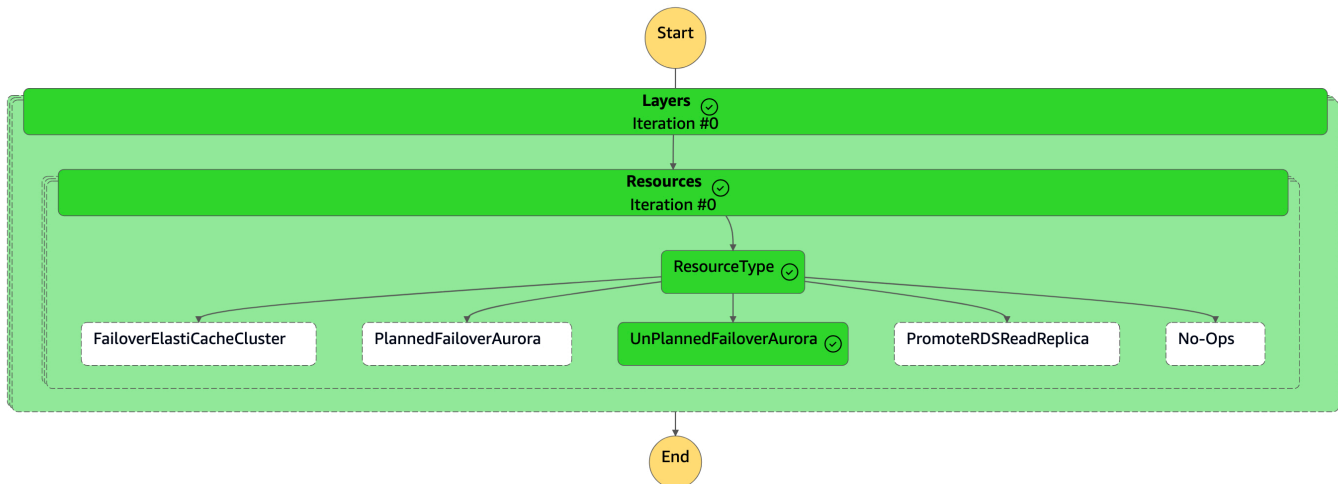
6. Die `dr-orchestrator-stepfunction-FAILOVER` Zustandsmaschine liest den Ressourcentyp als `UnPlannedFailoverAuroraMySQL` und ruft die Aufgabe `Detach Cluster from Global Database` von der `dr-orchestrator-stepfunction-unplanned-Aurora-failover` Zustandsmaschine aus auf.



7. Die Detach Cluster from Global Database Aufgabe trennt (entfernt) den sekundären Cluster von der globalen Datenbank.



8. Der sekundäre Cluster (dbcluster-02) wird zu einem eigenständigen Cluster heraufgestuft und kann read/write Workloads bedienen.
9. Die `dr-orchestrator-stepfunction-FAILOVER` Zustandsmaschine ist abgeschlossen.



10 Der sekundäre Cluster (dbcluster-02) ist von der globalen Aurora-Datenbank getrennt und wird zu einem eigenständigen Cluster, der die read/writer Arbeitslast bedient.

11 Konfigurieren Sie Ihre Anwendung neu, sodass alle Schreibvorgänge mithilfe des neuen Cluster-Endpunkts an diesen neuen eigenständigen Aurora-DB-Cluster gesendet werden.

Failback

Ein Failback bringt Ihre Datenbank an den ursprünglichen (oder neuen) primären Standort zurück, nachdem ein Notfall (oder ein geplantes Ereignis) behoben wurde. Wenn der ungeplante Ausfall behoben ist, möchten Sie möglicherweise Ihre frühere Hauptregion wieder zur globalen Aurora-Datenbank hinzufügen. Sie müssen zuerst den vorhandenen DB-Cluster aus der früheren primären Region löschen, einen neuen DB-Cluster aus der neuen primären Region erstellen und dann den verwalteten geplanten Failover-Prozess verwenden, um die Rolle des neuen Clusters zu wechseln.

Dies kann als geplante Aktivität betrachtet werden, die Sie außerhalb der Hauptverkehrszeiten oder an einem Wochenende durchführen können.

Sie müssen [den Amazon Aurora Aurora-DB-Cluster manuell ändern](#) und deaktivieren, DeletionProtection bevor Sie den DR Orchestrator FAILBACK Zustandsmaschine aus der früheren primären Region (us-east-1) ausführen, da er mit erstellt wurde DeletionProtection.

DR Orchestrator Framework verwendet die dr-orchestrator-stepfunction-FAILBACK Zustandsmaschine, um die Schritte zum Löschen des vorhandenen Clusters und zum Erstellen eines neuen Clusters in der ehemaligen Primärregion zu automatisieren.

Gehen Sie zum Deaktivieren DeletionProtection wie folgt vor:

1. Melden Sie sich bei der -Konsole an.
2. Ändern Sie die Region in die frühere Hauptregion (us-east-1).
3. Navigieren Sie zur Amazon RDS-Konsole, wählen Sie den Cluster-Namen (dbcluster-01) aus und klicken Sie auf Modify.
4. Deaktivieren Sie unter Löschschutz das Kontrollkästchen Löschschutz aktivieren und wählen Sie Weiter.
5. Wählen Sie Sofort anwenden und anschließend Cluster modifizieren aus.

Die DR Orchestrator FAILBACK Zustandsmaschine wird während des Failback-Vorgangs von der ehemaligen primären Region () us-east-1 aus aufgerufen.

Gehen Sie wie folgt vor, um das Failback durchzuführen:

1. Melden Sie sich bei der -Konsole an.
2. Ändern Sie die Region in die frühere primäre Region (us-east-1).
3. Navigieren Sie zu Services und wählen Sie dann Step Functions aus.
4. Navigieren Sie zur DR Orchestrator FAILBACK Zustandsmaschine.
5. Wählen Sie Ausführung starten und geben Sie den folgenden JSON-Code in den Input - optional Abschnitt ein:

```
{
  "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",

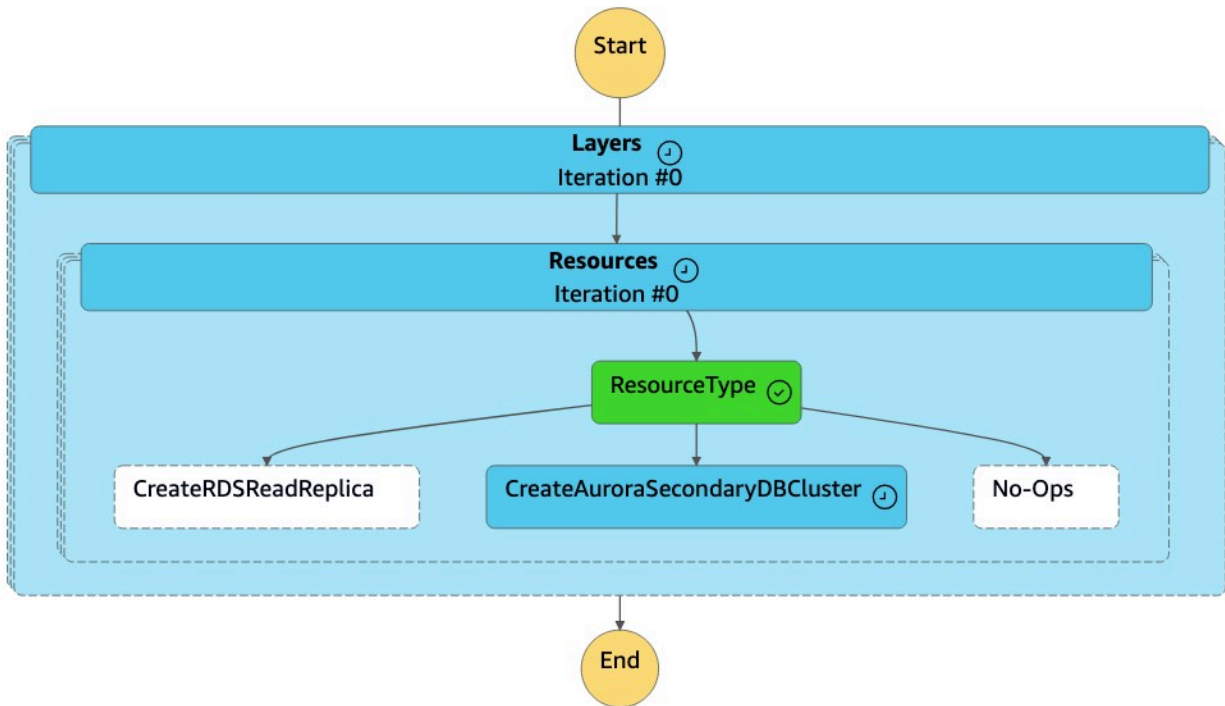
```

```

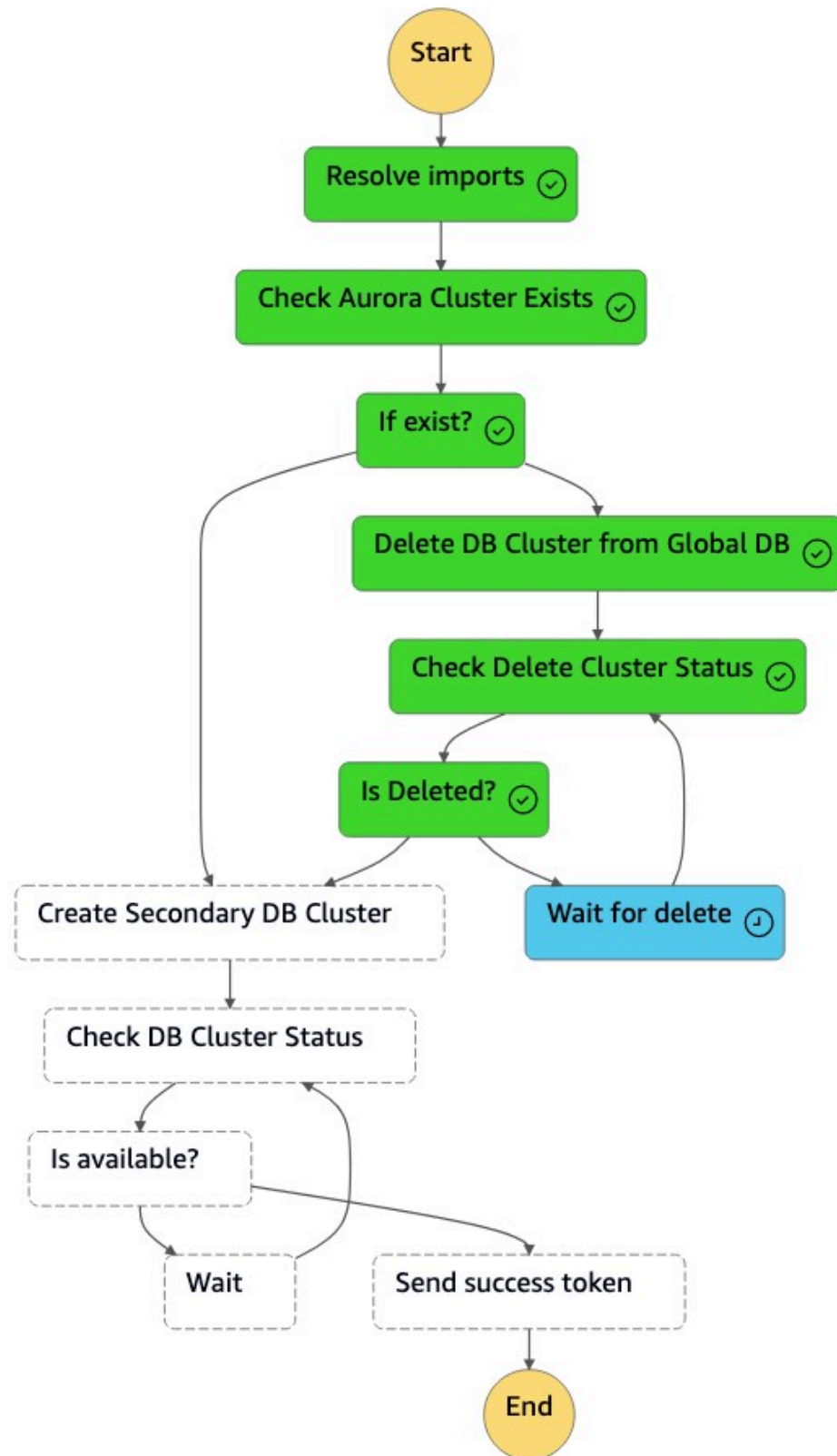
        "DBInstanceIdentifier": "!Import dr-globaldb-cluster-mysql-instance-
identifier",
        "Port": "!Import dr-globaldb-cluster-mysql-port",
        "DBInstanceClass": "!Import dr-globaldb-cluster-mysql-instance-class",
        "DBSubnetGroupName": "!Import dr-globaldb-cluster-mysql-subnet-group-
name",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

```

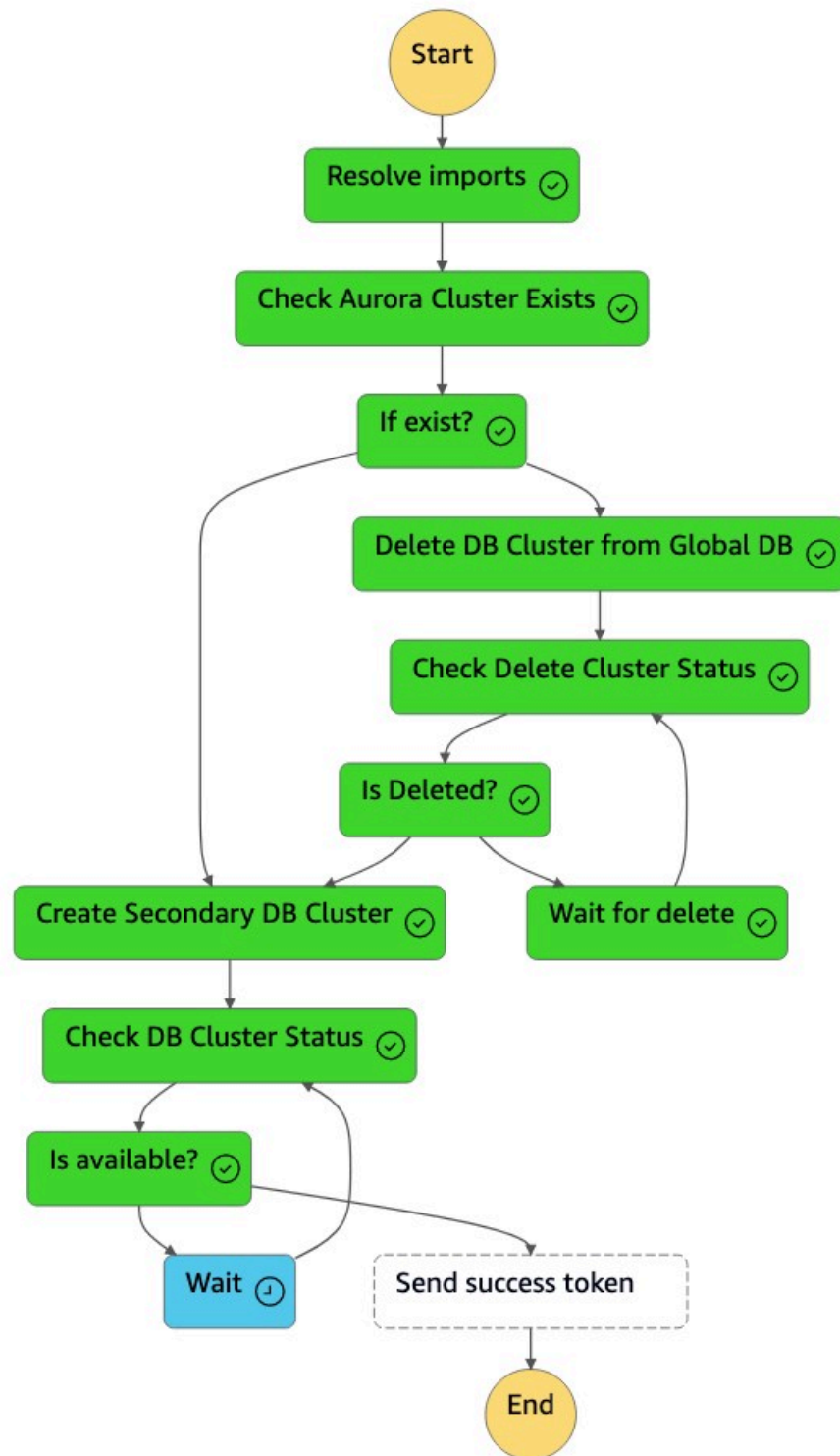
6. Die DR Orchestrator FAILBACK Zustandsmaschine liest den Ressourcentyp als `CreateAuroraSecondaryDBCluster` und ruft die `dr-orchestrator-stepfunction-create-Aurora-Secondary-cluster` Zustandsmaschine auf.



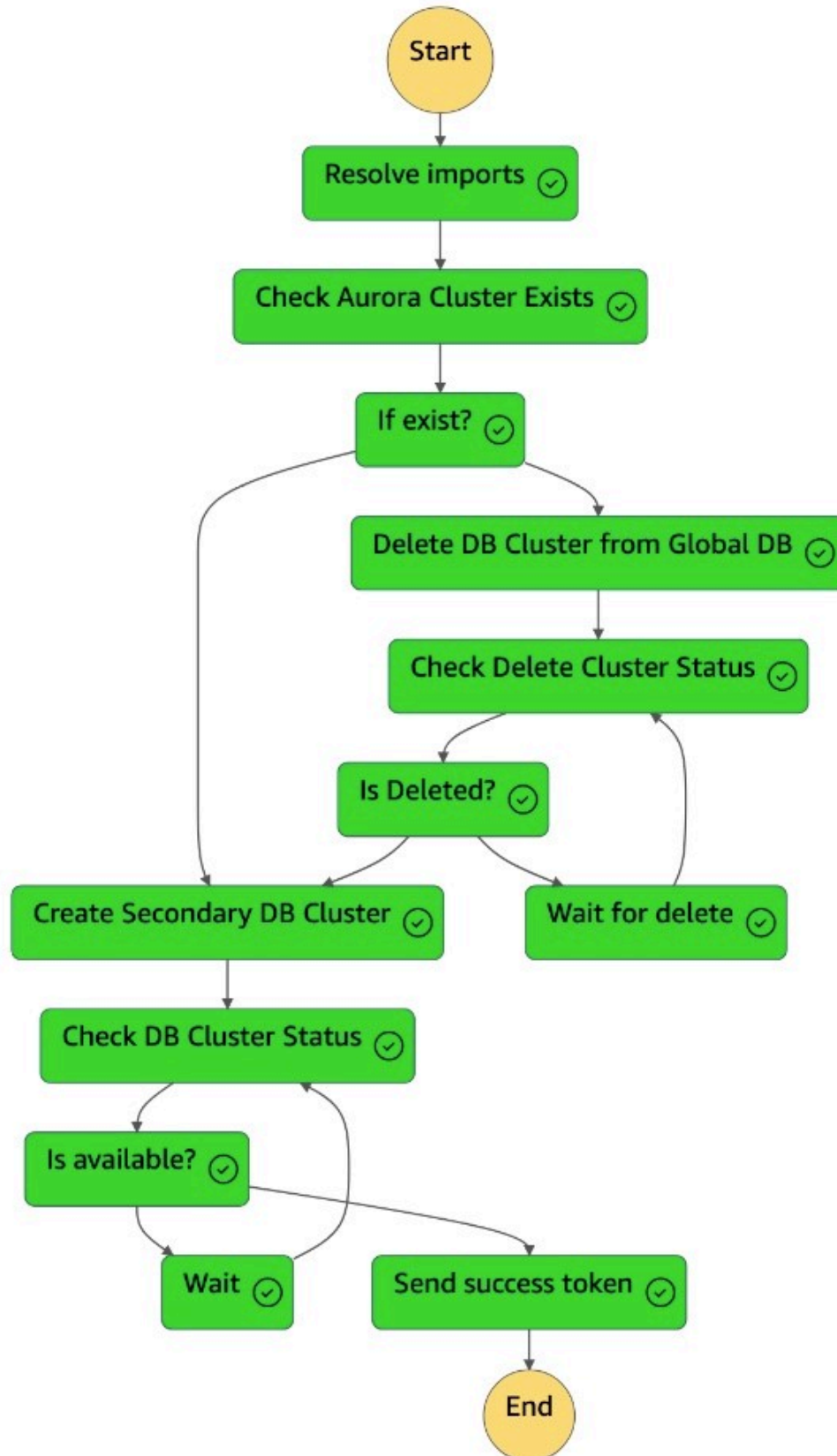
7. Die `dr-orchestrator-stepfunction-create-Aurora-Secondary-cluster` Zustandsmaschine löscht den vorhandenen Cluster (`dbcluster-01`) aus der früheren primären Region (`us-east-1`).



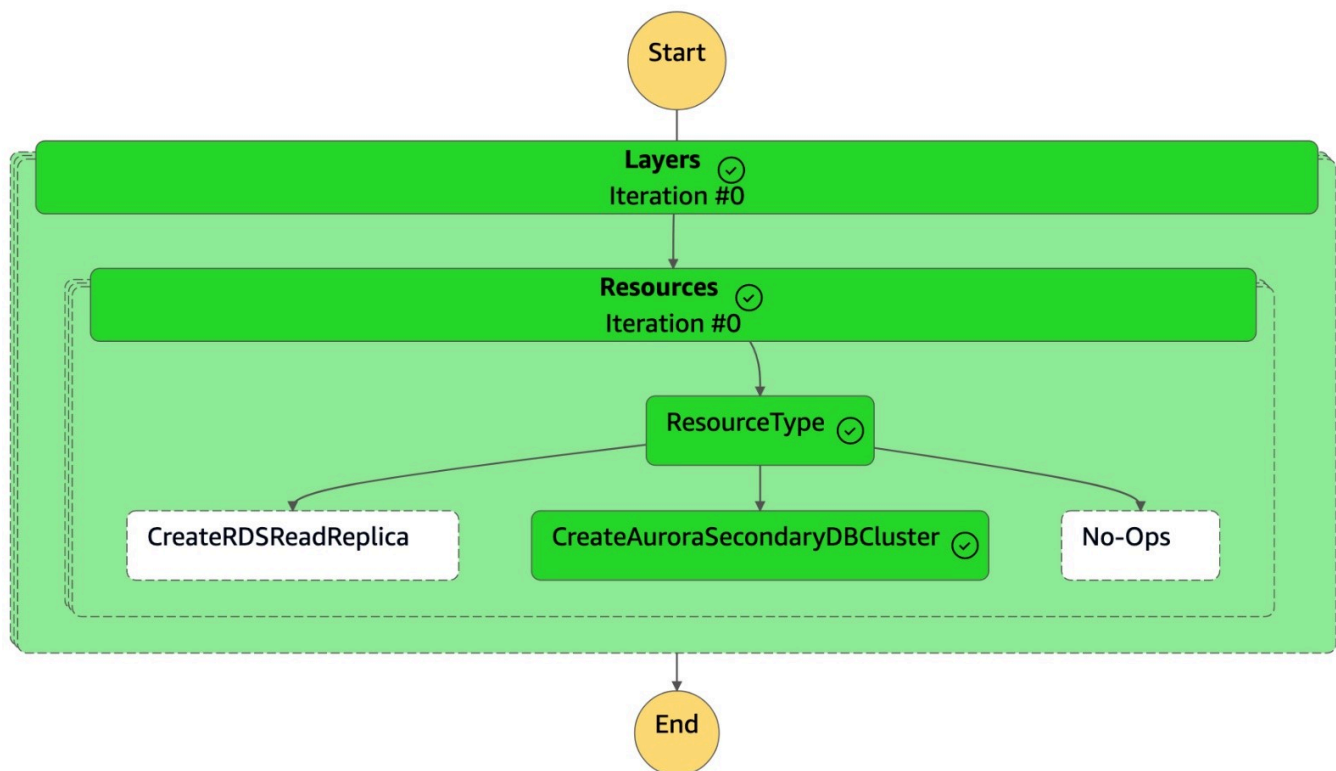
8. Nachdem der Cluster (dbcluster-01) gelöscht wurde, erstellt die Zustandsmaschine zusammen mit der DB-Instance einen neuen Cluster (dbcluster-01) und tritt der globalen Aurora-Datenbank als sekundärer Cluster bei, um schreibgeschützte Workloads zu bedienen.



9. Sobald der sekundäre Cluster verfügbar ist, ist die `dr-orchestrator-stepfunction-create-Aurora-Secondary-cluster` Zustandsmaschine fertig gestellt und sie sendet ein Erfolgstoken zurück an die DR Orchestrator Failback Zustandsmaschine.



10 Die dr-orchestrator-stepfunction-FAILBACK Zustandsmaschine ist abgeschlossen.



11 Sie können die globale Aurora-Datenbank auf der Amazon RDS-Konsole überprüfen.

Wenn Sie den primären DB-Cluster nach us-east-1 verlagern möchten, können Sie die im Abschnitt Switchover genannten Schritte ausführen.

Stellen Sie die Lösung bereit

Nachdem Sie die Voraussetzungen eingerichtet haben, folgen Sie den Anweisungen im [DR Orchestrator Framework-Muster](#). Anweisungen für Amazon und Amazon RDS Aurora sind enthalten ElastiCache. Um die Lösung zu testen, überprüfen Sie die Voraussetzungen und Einschränkungen und stellen Sie Beispieldatenbanken auf bereit AWS.

Voraussetzungen

- Aktiv [AWS-Konto](#) für die Bereitstellung dieser Lösung.
- Berechtigungen zum Erstellen von [Amazon Simple Storage Service \(Amazon S3\)](#) -Buckets und der in den [AWS CloudFormation](#) Vorlagen verwendeten Ressourcen.
- Bevor Sie die DR Orchestrator FAILBACK Zustandsmaschine ausführen:
 - [Ändern Sie die RDS Instance](#) manuell oder [ändern Sie den Amazon Aurora Aurora-DB-Cluster](#) und deaktivieren Sie ihn `DeletionProtection`.
 - Erstellen Sie vor dem Löschen der RDS Amazon-Instance oder des Aurora-Clusters einen [DB-Snapshot](#), wenn Sie eine Ursachenanalyse durchführen möchten (RCA).

Einschränkungen

- Die Verfügbarkeit und der Support der Funktionen variieren je nach Version der einzelnen Datenbank-Engines und zwischen den einzelnen Versionen AWS-Regionen. Weitere Informationen zur Verfügbarkeit von Funktionen und Regionen für die regionsübergreifende Replikation finden Sie unter [Regionsübergreifende Read Replicas](#).
- Diese Lösung wurde mit einer primären Datenbankinstanz mit einer Read Replica getestet. Wenn Sie mehr als eine Read Replica verwenden möchten, testen Sie die Lösung gründlich, bevor Sie sie in einer Produktionsumgebung implementieren.
- Für Amazon RDS unterstützt die Lösung nur die folgenden Engines: [Amazon RDS for My SQL](#), [Amazon RDS for Postgre SQL](#) und [Amazon RDS for MariaDB](#).

Stellen Sie Beispieldatenbanken bereit auf AWS

Um die regionsübergreifende DR-Lösung für Datenbanken zu demonstrieren AWS, können Sie [Amazon RDS for MySQL](#), [Amazon Aurora My SQL -Compatible Edition](#) und [Amazon ElastiCache \(RedisOSS\)](#) — Global Datastore bereitstellen. [Folgen Sie den Anweisungen in der App-](#)

[StackREADME Datei des -databases-Repositorys. aws-cross-region-dr](#) GitHub In diesem Beispiel ist us-east-1 die primäre Region und die sekundäre Region. us-west-2

Der Anwendungsstapel verwendet den Abschnitt AWS CloudFormation [Outputs](#), um die Parameter zu exportieren, die vom DR Orchestrator Framework zur Automatisierung von Failover und Failback von Datenbanken verwendet werden. AWS

Häufig gestellte Fragen

Hier finden Sie Antworten auf Fragen zum DR Orchestrator Framework für Failover und Failback.

Welche RPO- und RTO-Werte kann ich mit diesem Ansatz erreichen?

Informationen zu Recovery Time Objective (RTO) und Recovery Point Objective (RPO) finden Sie in der Strategie [Disaster Recovery Strategy for Databases](#) auf AWS.

Ist die Verwendung von AWS CloudFormation Exportvariablen obligatorisch?

Nein, Sie können den Wert der globalen Amazon Aurora Datenbank oder der Amazon RDS-DB-Instance direkt im JSON-Format übergeben (zum Beispiel: - "RDSInstanceIdentifier": "rds-mysql-instance").

Kann ich den DR Orchestrator FAILOVER-Workflow verwenden, um ein Failover für mehr als eine AWS Datenbank durchzuführen?

Ja, Sie können mehr als eine Ressource in der Eingabeparameterdatei übergeben, um ein Failover für mehr als eine AWS Datenbank durchzuführen. Das folgende Codebeispiel zeigt ein paralleles Failover einer Amazon RDS for MySQL MySQL-Read Replica und eines globalen Amazon-Datenspeichers ElastiCache (Redis OSS):

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

```

Wie kann ich den InvalidParameterCombination Fehler vermeiden, wenn ich die DR Orchestrator FAILBACK State Machine für Amazon RDS ausführe?

Der vollständige Text des Fehlers lautet:

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

Um den Fehler zu vermeiden, [ändern Sie die RDS-Instanz](#), indem Sie DeletionProtection sie deaktivieren, bevor Sie den DR Orchestrator FAILBACK State Machine ausführen.

Nächste Schritte

Sie können DR Orchestrator Framework verwenden, um einen Genehmigungsprozess hinzuzufügen, bevor Sie die Failover- oder Failback-Aktivitäten einleiten. Wenn Sie beispielsweise den Failover während eines DR-Ereignisses einleiten, können Sie zusätzlich eine Zustandsmaschine hinzufügen, DR Orchestrator Failover um eine [Amazon Simple Notification Service \(Amazon SNS\) - Benachrichtigung](#) per E-Mail zu senden. Sobald die Genehmigung erteilt wurde, beginnt die Failover-Aktivität.

Das Anwendungs-Failover kann auch in das DR Orchestrator Framework integriert werden, um den DNS-Eintrag umzuschalten und Anwendungen auf den neuen Endpunkt der Datenbankinstanz oder des Clusters zu verweisen.

Sie können die [AWS X-Ray](#) Ablaufverfolgung verwenden, um die Failover-Dauer für die Berechnung des RTO zu ermitteln. Sie können ein Monitoring-Dashboard auf X-Ray aufbauen.

Ressourcen

- [Replikation AWS-Regionen über globale Datenspeicher hinweg](#) (Amazon ElastiCache (Redis OSS) -Dokumentation)
- [Verwenden von Failover in einer Amazon Aurora Global Database](#) (Aurora-Dokumentation)
- [Automatisieren Sie das Amazon Aurora Global Database-Endpunktmanagement für geplante und ungeplante Failover](#) (Blogbeitrag)
- [Notfallwiederherstellung: 3 Failover-Szenarien für Ihre globale Amazon Aurora Aurora-Datenbank mit Terraform \(Teil 2\)](#) (Blogbeitrag)
- [Notfallwiederherstellungsstrategie für Datenbanken auf AWS](#) (AWS Prescriptive Guidance Strategy)
- [Automatisieren Sie regionsübergreifendes Failover und Failback mithilfe des DR Orchestrator Framework](#) (Prescriptive Guidance Pattern)AWS

Dokumentverlauf

In der folgenden Tabelle werden wichtige Änderungen in diesem Leitfaden beschrieben. Um Benachrichtigungen über zukünftige Aktualisierungen zu erhalten, können Sie einen [RSS-Feed](#) abonnieren.

Änderung	Beschreibung	Datum
Erste Veröffentlichung	—	3. Mai 2024

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.