



에서 관계형 데이터베이스에 대한 DR 솔루션 자동화 AWS

AWS 권장 가이드



AWS 권장 가이드: 에서 관계형 데이터베이스에 대한 DR 솔루션 자동화 AWS

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

Amazon의 상표 및 트레이드 드레스는 Amazon 외 제품 또는 서비스와 함께, Amazon 브랜드 이미지를 떨어뜨리거나 고객에게 혼동을 일으킬 수 있는 방식으로 사용할 수 없습니다. Amazon이 소유하지 않은 기타 모든 상표는 Amazon과 제휴 관계이거나 관련이 있거나 후원 관계와 관계없이 해당 소유자의 자산입니다.

Table of Contents

소개	1
목표	1
개요	2
장애 조치 프로세스	2
DR Orchestrator FAILOVER 아키텍처	3
장애 복구 프로세스	4
DR Orchestrator FAILBACK 아키텍처	4
DR 오케스트레이터 장애 조치	6
파라미터 세부 정보	7
DR 오케스트레이터 장애 조치	8
Aurora 상태 및 Step Functions 상태 시스템	11
정상 상태	11
이벤트 상태	11
전환	12
계획되지 않은 수동 장애 조치	17
장애 복구	21
솔루션 배포	30
FAQ	32
이 접근 방식을 사용하여 달성할 수 있는 RPO 및 RTO는 무엇입니까?	32
AWS CloudFormation 내보내기 변수를 사용해야 합니까?	32
DR Orchestrator FAILOVER 워크플로를 사용하여 둘 이상의 AWS 데이터베이스를 장애 조치할 수 있습니까?	32
Amazon RDS용 DR Orchestrator FAILBACK 상태 시스템을 실행할 때 InvalidParameterCombination 오류를 방지하려면 어떻게 해야 합니까?	33
다음 단계	34
리소스	35
사용 설명서 기록	36
.....	xxxvii

에서 관계형 데이터베이스에 대한 DR 솔루션 자동화 AWS

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

2024년 5월([문서 기록](#))

Amazon Web Services([AWS](#))에서 [재해 복구\(DR\) 전략](#)을 계획AWS할 때 조직의 데이터베이스에 대한 DR 솔루션을 구현할 수 있습니다. 데이터베이스 인스턴스의 장애 조치를 수행하도록 이벤트 기반 아키텍처를 구성하여 DR 솔루션을 자동화할 수 있습니다. 이 접근 방식의 이점은 다음과 같습니다.

- 수동 조정으로 인한 인적 오류 감소
- 경우에 따라 가능한 가장 짧은 복구 시간 목표(RTO) 제공
- DR 솔루션의 반복 가능한 테스트 촉진

자동화가 없으면 기본 중단 AWS 리전으로 인해 조직의 리소스가 런북에서 단계를 수동으로 수행해야 합니다. 이러한 단계에는 새로운 기본 데이터베이스 인스턴스 역할을 하도록 보조의 Amazon Relational Database Service(RDS) 읽기 전용 복제본을 승격하는 등 여러 프로세스가 포함될 수 있습니다. 이러한 작업을 자동화하기 [AWS 위해 Python용 SDK\(Boto3\)](#) APIs를 AWS 제공하며, 이는 조직의 필요에 맞는 DR 솔루션의 구성 요소를 구성할 수 있습니다. AWS 리전

이 가이드에서는 동일한 AWS 리전 내의에서 AWS 데이터베이스에 대한 DR 솔루션을 달성하는 데 도움이 될 수 있는 중앙 집중식 매니페스트 기반 DR 오케스트레이터에 대해 설명합니다 AWS 계정. 장애 조치 수행 결정은 여전히 사람의 개입이 필요하며 자동화 범위를 벗어납니다.

이 가이드의 대상은 애플리케이션 아키텍트, 인프라 아키텍트, 데이터베이스 아키텍트 및 데이터베이스 컨설턴트입니다.

목표

DR Orchestrator Framework 솔루션을 구현하면 다음과 같은 결과를 얻을 수 있습니다.

- 중단 중 수동 오류 감소 - DR 프로세스의 수동 성능에 오류가 발생하기 쉽습니다. DR 활동 단계에서 작업을 자동화하면 수동 오류가 줄어들고 장애 조치 또는 장애 복구가 더 원활하게 이루어집니다.
- 대규모 DR 관리 - 조직에 대규모 데이터베이스 집합이 있는 경우 DR 자동화를 구축하면 DR 솔루션을 확장하는 데 도움이 됩니다. DR Orchestrator Framework를 사용하여 AWS 데이터베이스 DR 솔루션을 자동화할 수 있습니다. DR Orchestrator는 JSON 형식의 파라미터 파일을 사용하여 AWS 데이터베이스의 장애 조치 또는 장애 복구 단계를 자동화합니다.

DR 오케스트레이터 프레임워크 개요

DR Orchestrator Framework는 AWS 데이터베이스의 리전 간 DR을 오케스트레이션하고 자동화하는 원클릭 솔루션을 제공합니다. [AWS Step Functions](#) 및 [AWS Lambda](#)를 사용하여 장애 조치 및 장애 복구 중에 필요한 단계를 수행합니다. Step Functions 상태 시스템은 오케스트레이터 설계 내에서 의사 결정을 위한 기반을 제공합니다. 장애 조치 또는 장애 복구 작업을 수행하기 위한 API 작업은 상태 시스템 내에서 호출되는 Lambda 함수로 코딩됩니다. Lambda 함수는 [AWS SDK for Python \(Boto3\)](#) APIs하여 AWS 데이터베이스와 상호 작용합니다.

DR Orchestrator Framework에는 장애 조치 및 장애 복구 단계에 해당하는 두 개의 기본 상태 시스템이 포함되어 있습니다.

Amazon RDS의 경우 장애 조치 단계에서는 리전 간 RDS 읽기 전용 복제본을 독립 실행형 DB 인스턴스로 승격합니다. Amazon Aurora의 경우 드물지만 예기치 않은 중단 중에 기본 리전이 다운되면 라이터 노드를 사용할 수 없습니다. 라이터 노드와 보조 클러스터 간의 복제가 중지됩니다. 글로벌 데이터베이스에서 보조 클러스터를 분리하고 독립 실행형 클러스터로 승격해야 합니다. 애플리케이션은 쓰기 트래픽을 연결하여 독립 실행형 클러스터로 전송할 수 있습니다. 이 동일한 프로세스를 사용하여 글로벌 데이터베이스의 기본 DB 클러스터를 보조 리전으로 [전환할](#) 수 있습니다. 다음과 같은 제어된 시나리오에는이 접근 방식을 사용합니다.

- 운영 유지 관리
- 계획된 운영 절차
- Amazon ElastiCache(Redis OSS) 보조 클러스터를 새 기본 클러스터로 승격

장애 복구 단계에서는 라이브 기본 리전과 새 보조 리전 간에 데이터의 라이브 복제를 설정합니다.

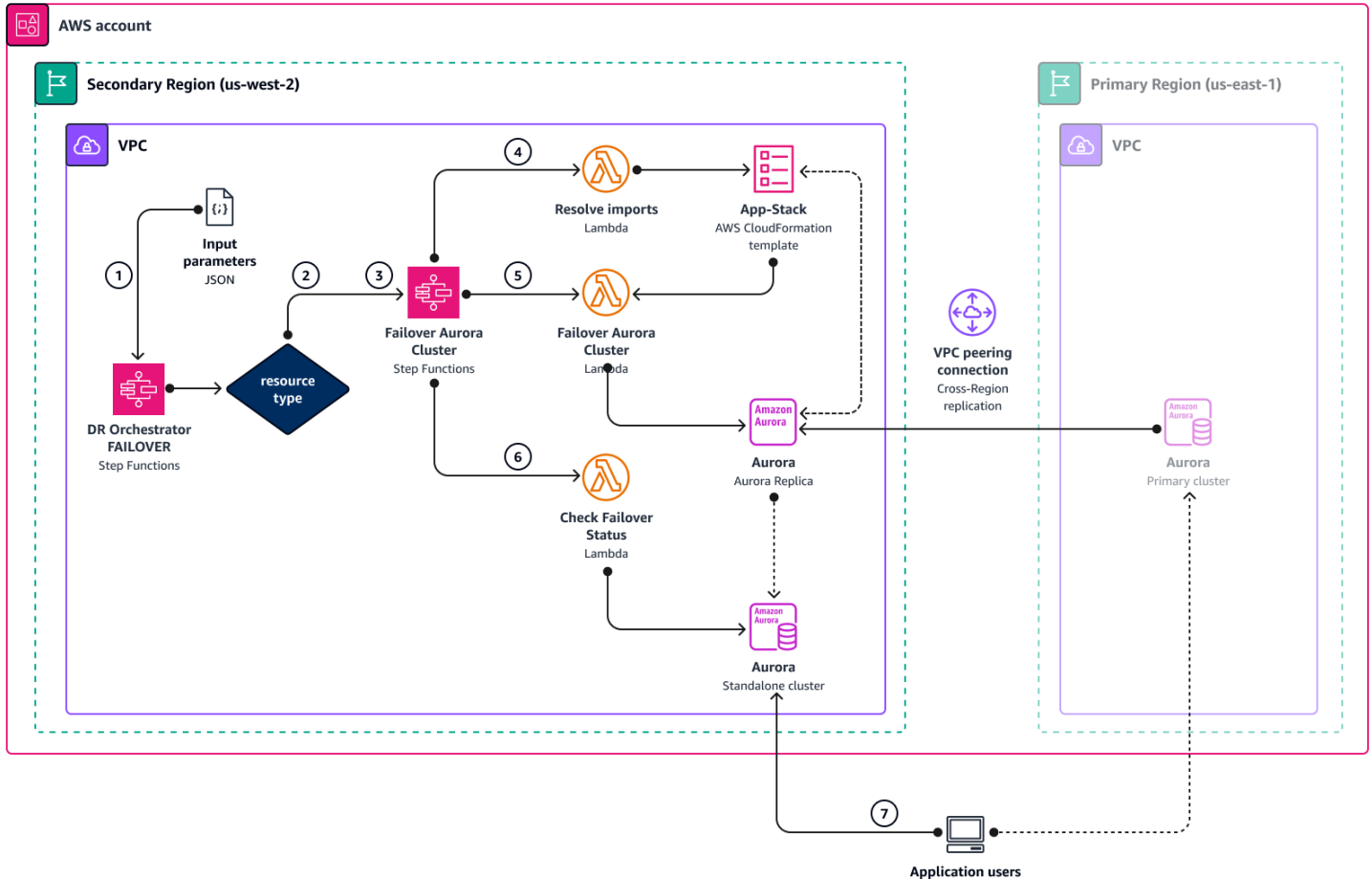
DR Orchestrator는 데이터베이스에만 적용된다는 점을 이해하는 것이 중요합니다. 이러한 데이터베이스를 참조하고 동일한 리전에 있는 모든 애플리케이션에는 별도의 탠덤 장애 조치 솔루션이 필요할 수 있습니다. 데이터베이스가 보조 리전으로 장애 조치되면 새 데이터베이스 인스턴스에 연결하도록 애플리케이션을 업데이트해야 합니다. 새 데이터베이스 인스턴스는 데이터 소스 역할을 합니다.

장애 조치 프로세스

장애 조치를 수행하려면 DR Orchestrator FAILOVER 상태 시스템을 실행합니다. 이 단계에서 보조 데이터베이스는 이미 보조 리전에 읽기 전용 복제본(Amazon RDS) 또는 보조 클러스터(Amazon Aurora)로 존재합니다. DR Orchestrator FAILOVER 상태 시스템을 실행하면 보조 데이터베이스가 기본 데이터베이스로 승격됩니다.

DR Orchestrator FAILOVER 아키텍처

다음 다이어그램은 DR Orchestrator를 사용할 때 Amazon Aurora에 대한 장애 조치 프로세스의 개념을 보여줍니다. Amazon Aurora와 Amazon ElastiCache는 동일한 워크플로를 사용하지만 상태 시스템과 Lambda 함수는 다릅니다.



1. DR Orchestrator FAILOVER 상태 시스템은 입력 JSON 파라미터를 읽습니다.
2. 상태 시스템은 resourceType 파라미터를 기반으로, Promote RDS Read Replica Failover Aurora Cluster 또는 등 다른 상태 시스템을 호출합니다. 둘 이상의 리소스가 입력에 전달되면 이러한 상태 머신이 병렬로 실행됩니다.
3. Failover Aurora Cluster 상태 시스템은 다음 세 단계 각각에서 Lambda 함수를 호출합니다.
4. Resolve imports Lambda 함수는 App-Stack AWS CloudFormation 템플릿 "import <export-variable-name>"의 실제 값으로 확인됩니다.
5. Lambda 함수는 Failover Aurora Cluster 읽기 전용 복제본을 독립 실행형 DB 인스턴스로 승격합니다.

6. Lambda 함수는 Check Failover Status 승격된 DB 인스턴스의 상태를 확인합니다. 상태가 AVAILABLE이면 Lambda 함수는 성공 토큰을 호출 상태 시스템으로 다시 전송하고 완료합니다.
7. 애플리케이션을 이제 기본 데이터베이스인 DR 리전(us-west-2)의 독립 실행형 데이터베이스로 리디렉션할 수 있습니다.

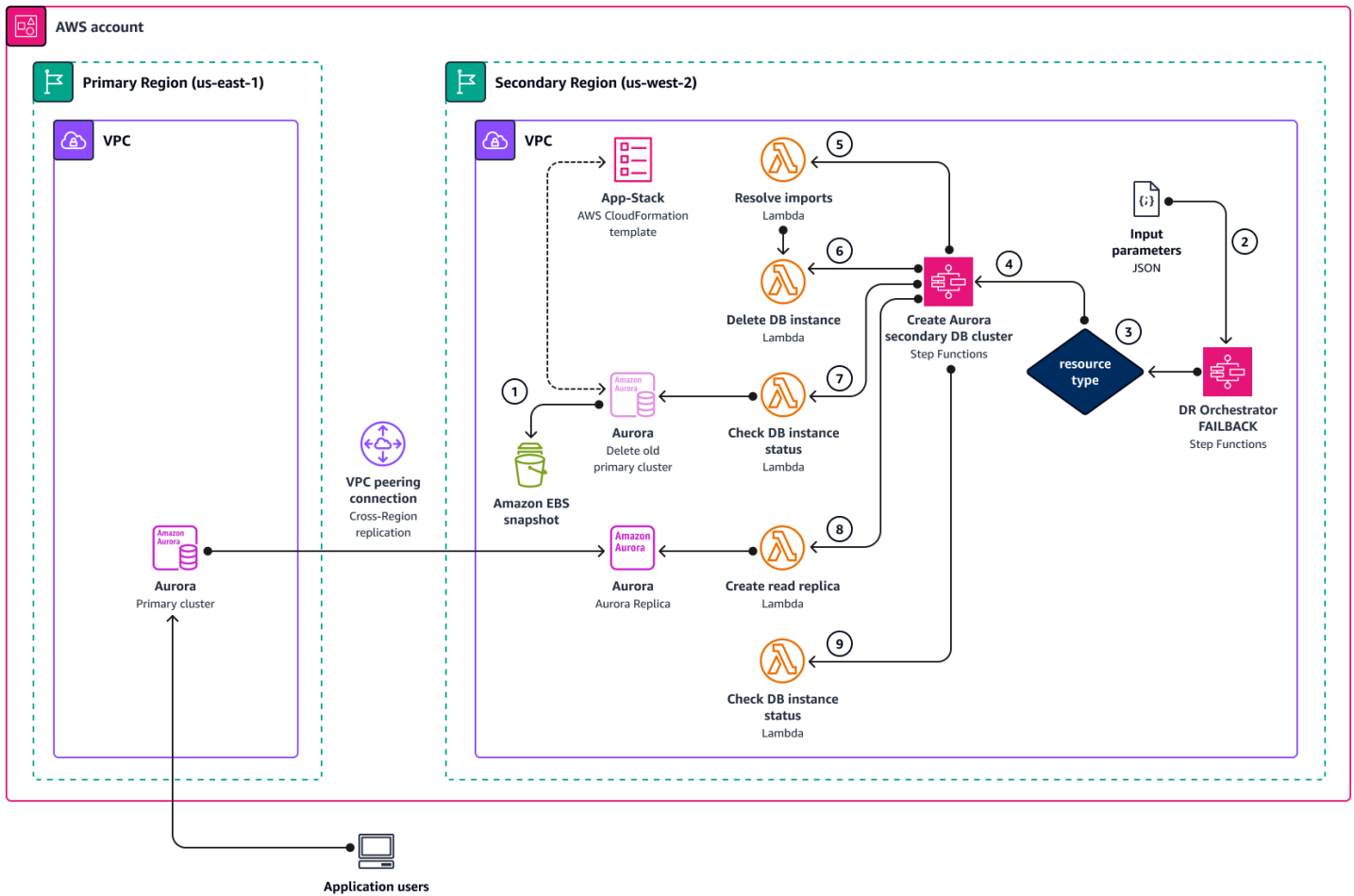
장애 복구 프로세스

이전 기본 리전(us-east-1)이 다시 가동되면 다시 실패하여의 데이터베이스가 다시 기본 리전us-east-1이 될 수 있습니다. 장애 복구를 시작하려면 DR Orchestrator FAILBACK 상태 시스템을 실행합니다. 이름에서 알 수 있듯이이 상태 시스템은 새 기본 리전(us-west-2)의 변경 사항을 현재 보조 리전 역할을 하는 이전 기본 리전(us-east-1)으로 다시 복제하기 시작합니다.

두 리전 간에 복제가 설정되면 장애 복구를 시작할 수 있습니다. 페일백하고 원래 기본 리전(us-east-1)으로 돌아가려면 현재 보조 리전(us-east-1)에서 DR Orchestrator FAILOVER 상태 시스템을 실행하여 기본 리전으로 승격합니다.

DR Orchestrator FAILBACK 아키텍처

다음 다이어그램은 DR Orchestrator를 사용할 때 Amazon Aurora에 대한 페일백 프로세스의 개념을 보여줍니다.



1. 장애 복구를 시작하기 전에 근본 원인 분석(RCA)을 수행할 때 사용할 수동 DB 스냅샷을 생성합니다.

또한 이전 기본 리전(us-east-1)의 Aurora 클러스터에 DeletionProtection 상태를 비활성화합니다.

2. DR Orchestrator FAILBACK 상태 시스템은 입력 JSON 파라미터를 읽습니다.

3. resourceType DR Orchestrator FAILBACK 상태 시스템은 Create Aurora Secondary DB Cluster 상태를 호출합니다.

4. Create Aurora Secondary DB Cluster 상태 시스템은 다음 5단계 각각에서 Lambda 함수를 호출합니다.

5. Resolve import Lambda 함수는 App-Stack CloudFormation 템플릿 "! import <export-variable-name>"의 실제 값으로 확인됩니다.

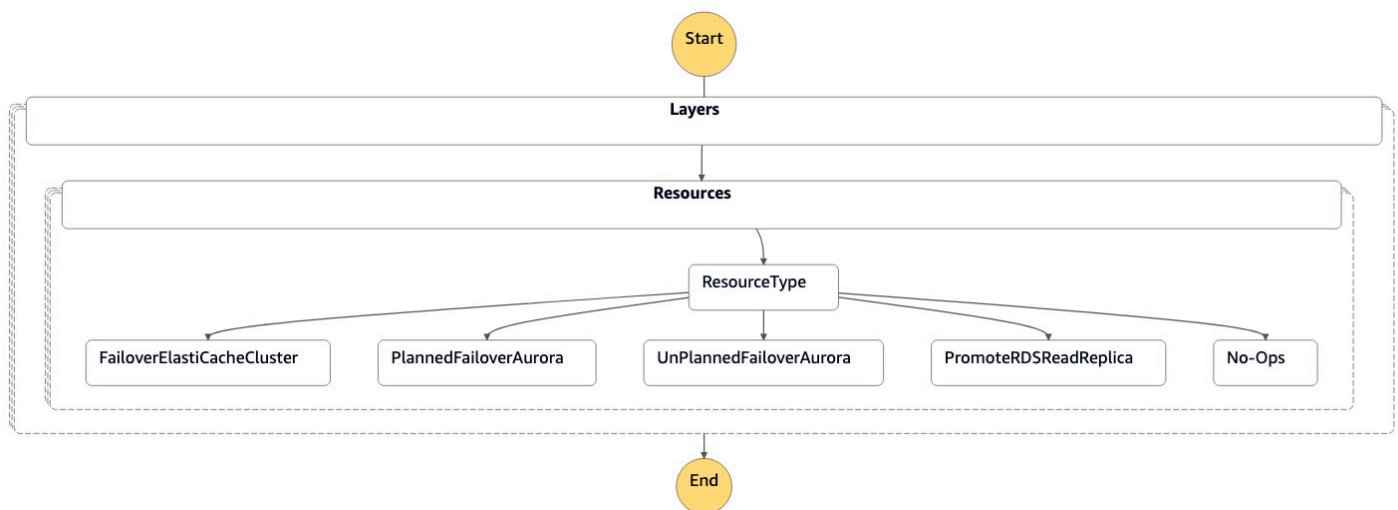
6. Delete DB Instance Lambda 함수는 이전 기본 인스턴스를 삭제합니다.

7. Check DB instance status Lambda 함수는 DB가 삭제 Delete DB Instance status될 때까지를 확인합니다.
8. Create Read Replica Lambda 함수는 새 기본 리전에 있는 DB 인스턴스에서 보조 리전에 읽기 전용 복제본을 생성합니다.
9. Check DB instance status Lambda 함수는 읽기 전용 복제본 DB 인스턴스 상태를 확인합니다. 상태가 AVAILABLE이면 Lambda 함수는 성공 토큰을 호출 상태 머신으로 다시 전송하고이 시스템은 완료됩니다.

DR 오케스트레이터 장애 조치

기본 리전(us-east-1)이 다운되거나 운영 유지 관리와 같은 계획된 이벤트 중에 DR 이벤트에서 DR Orchestrator FAILOVER 상태 시스템을 사용합니다.

함수를 호출하여 단일 또는 여러 데이터베이스를 병렬로 장애 조치할 수 있습니다.



상태 시스템은 다음 코드와 같이 JSON 형식의 파라미터를 수락합니다.

```

{
  "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"
        }
    }
]
}
]
}

```

파라미터 세부 정보

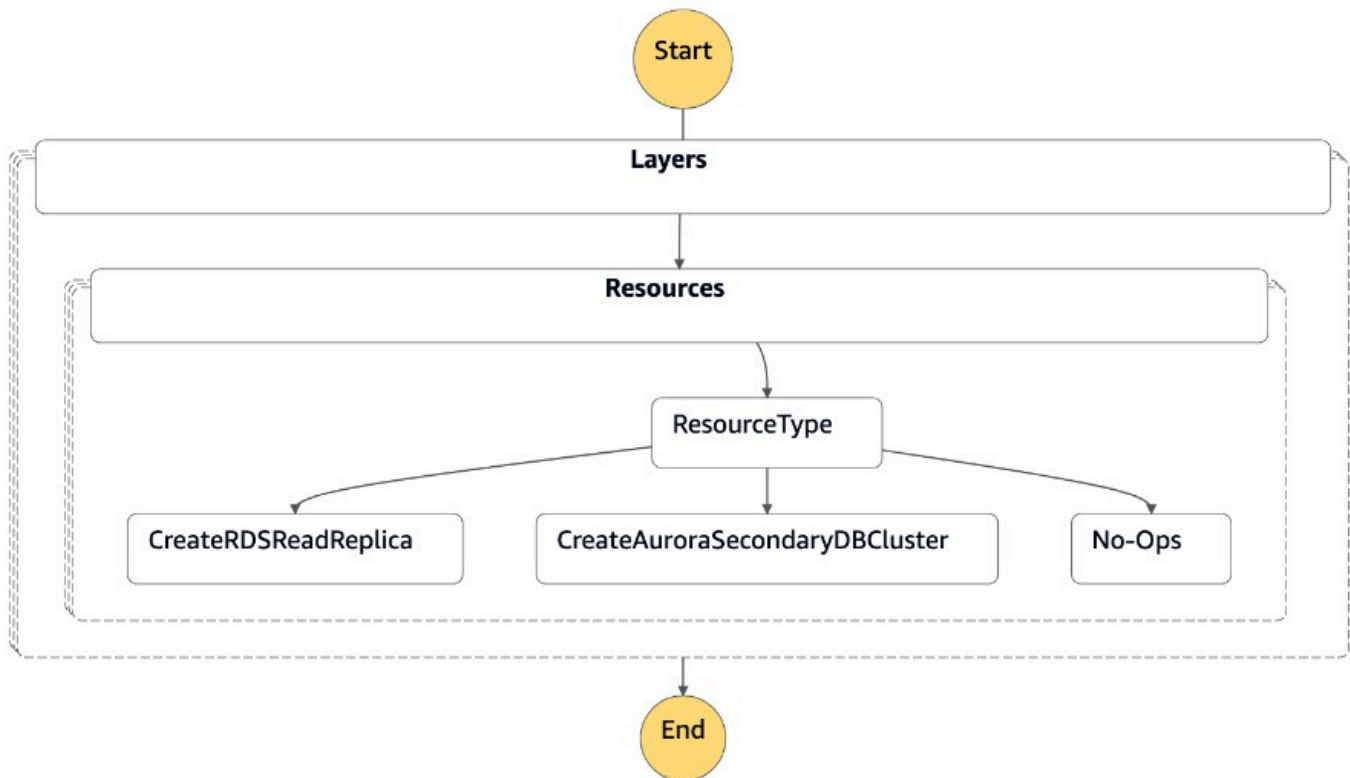
다음 표에는 DR Orchestrator FAILOVER 상태 시스템에서 사용하는 파라미터가 나와 있습니다.

파라미터 이름	설명	예상 값
layer (필수: 숫자)	처리 시퀀스입니다. 계층 2 리소스를 실행하기 전에 계층 1에 정의된 모든 리소스를 실행해야 합니다.	1 또는 2 등
리소스(필수: 사전 배열)	단일 계층 내의 모든 리소스는 병렬로 실행됩니다.	<pre> { "resourceType": " String", "resourceName": "St ring", "parameters": { "<param1>": "<!Import cft-output-1">, } </pre>

파라미터 이름	설명	예상 값
resourceType (필수: 문자열)	리소스를 식별할 리소스 유형	PromoteRDSReadReplica 또는 FailoverElastiCacheCluster
resourceName (선택 사항: 문자열)	이러한 리소스가 속한 애플리케이션 포트폴리오를 식별하려면	Promote RDS for MySQL Read Replica
파라미터(필수: 사전 배열)	데이터베이스를 장애 조치하거나 장애 복구 AWS 하는 데 필요한 파라미터 목록	<pre>{ "<param1>": "<!Import cft- output-1>", "<param2>": "<!Import cft- output-2>", }</pre>

DR 오케스트레이터 장애 조치

이전 기본 리전(us-east-1)이 가동되면 DR 이벤트 후 DR Orchestrator FAILBACK 상태 시스템을 사용합니다. 새 기본 리전(us-west-2)의 이전 기본 리전에서 Amazon RDS에 대한 [읽기 전용 복제본](#)을 생성하여 DR 전략을 준수할 수 있습니다. 계획된 이벤트이므로 주말 또는 사용량이 적은 업무 시간에이 활동을 예약하여 예상 가동 중지 시간을 확인할 수 있습니다.



상태 시스템은 다음 코드와 같이 JSON 형식의 파라미터를 수락합니다.

```

{
  "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 상태 및 Step Functions 상태 시스템

이 섹션에서는 Amazon Aurora 클러스터 장애 조치 및 장애 복구와 관련된 프로세스 및 상태 시스템을 다룹니다. 클러스터는 글로벌 데이터베이스로 구성됩니다.

Note

이 예제에서는 데모용으로 Aurora MySQL 호환 버전을 사용합니다. Aurora PostgreSQL 호환 버전에 유사한 단계를 사용할 수 있습니다.

정상 상태

정상 상태에서 Amazon Aurora MySQL 호환 글로벌 데이터베이스(dr-globaldb-cluster-mysql)는 두 개의 DB 클러스터로 생성되었습니다. 첫 번째 DB 클러스터(db-cluster-01)는 읽기/쓰기 워크로드를 제공하기 위해 기본 AWS 리전(us-east-1)에 생성되었습니다. 두 번째 DB 클러스터(db-cluster-02)는 읽기 전용 워크로드를 서버하기 위해 보조 리전(us-west-2)에 생성되었습니다.

DR 솔루션을 제공하는 것 외에도 애플리케이션에서 보조 DB 클러스터로 읽기 쿼리를 라우팅하여 기본 DB 클러스터의 부하를 줄일 수 있습니다. 이러한 각 클러스터에는 dbcluster-01-use1-instance-1 및 라는 데이터베이스 인스턴스가 dbcluster-02-usw2-instance-2 각각 하나씩 포함되어 있습니다.

이벤트 상태

Amazon Aurora 글로벌 데이터베이스를 사용하면 재해를 매우 빠르게 계획하고 복구할 수 있습니다. 재해 복구는 일반적으로 Recovery Time Objective(RTO) 및 Recovery Point Objective(RPO) 값을 사용하여 측정됩니다. 자세한 내용은 [Amazon Aurora 글로벌 데이터베이스에서 전환 또는 장애 조치 사용을 참조하세요](#).

Aurora 글로벌 데이터베이스를 사용하면 장애 조치에 대한 두 가지 접근 방식이 있습니다.

- 전환(계획된 관리형 장애 조치)
- 장애 조치(계획되지 않은 수동 장애 조치 또는 분리 및 승격)

전환

전환은 운영 유지 관리 및 기타 계획된 운영 절차와 같은 제어된 환경을 위한 것입니다. 계획된 관리형 장애 조치를 사용하면 Aurora 글로벌 데이터베이스의 기본 DB 클러스터를 보조 리전 중 하나로 재배포할 수 있습니다. 전환은 보조 DB 클러스터가 기본 데이터베이스와 동기화될 때까지 대기하므로 RPO는 0(데이터 손실 없음)입니다. 자세한 내용은 [Amazon Aurora 글로벌 데이터베이스에 대한 전환 수행을 참조하세요](#).

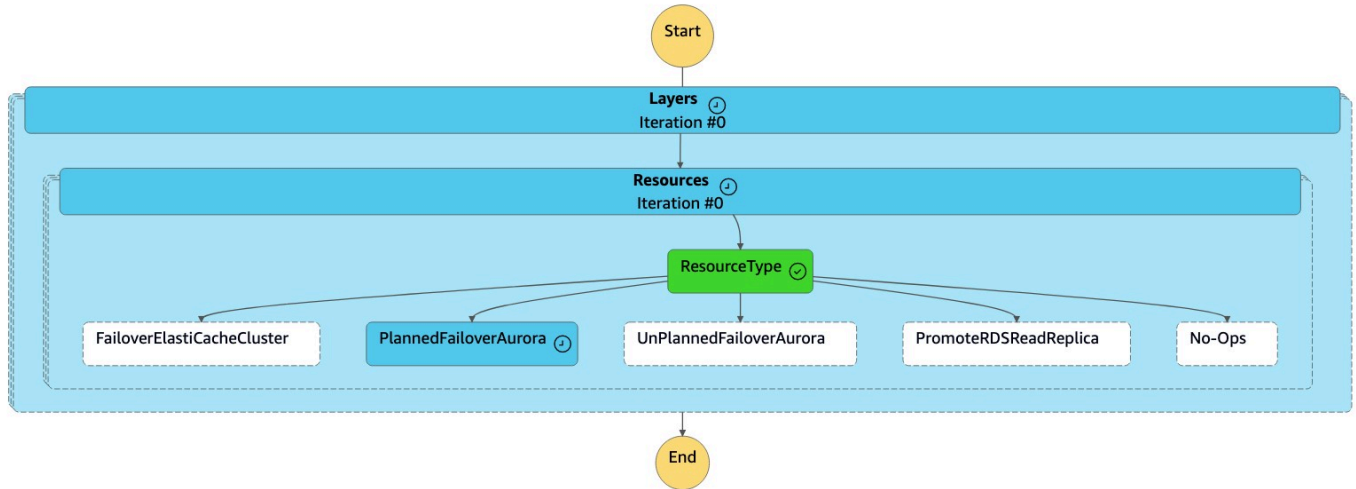
이벤트 상태 중에 dr-orchestrator-stepfunction-FAILOVER 상태 시스템이 호출되어 기본 클러스터를 선택한 보조 리전(us-west-2)으로 전환합니다.

전환을 수행하려면 다음을 수행합니다.

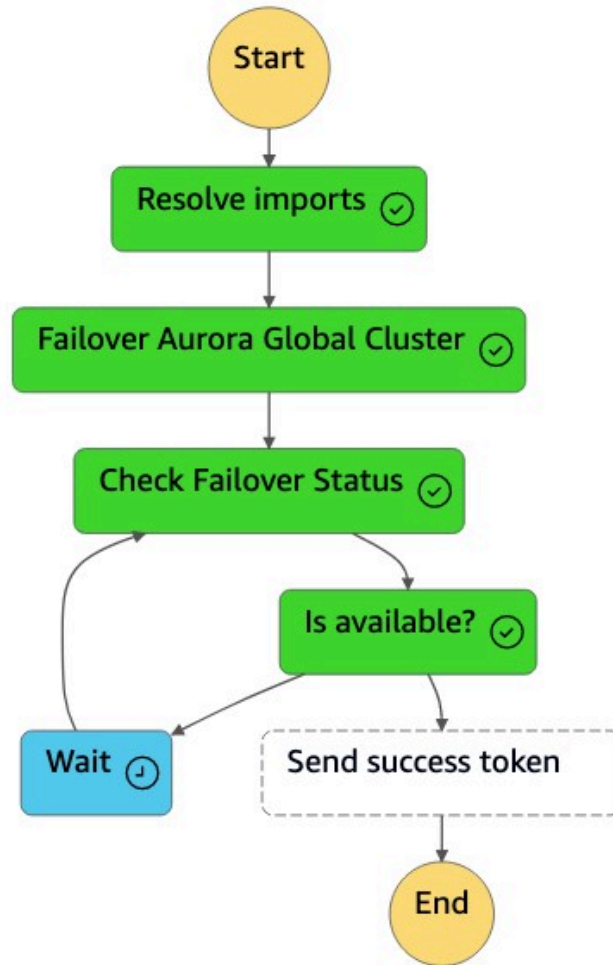
1. AWS Management Console에 로그인합니다.
2. 리전을 DR 리전(us-west-2)으로 변경합니다.
3. 서비스로 이동하여 Step Functions를 선택합니다.
4. dr-orchestrator-stepfunction-FAILOVER 상태 시스템으로 이동합니다.
5. 실행 시작을 선택하고 Input - optional 섹션에 다음 JSON 코드를 입력합니다.

```
{
  "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 상태 시스템은 리소스 유형을 PlannedFailoverAuroraMySQL로 읽고 dr-orchestrator-stepfunction-planned-Aurora-failover 상태 시스템을 호출하여 Aurora 글로벌 데이터베이스를 장애 조치합니다.



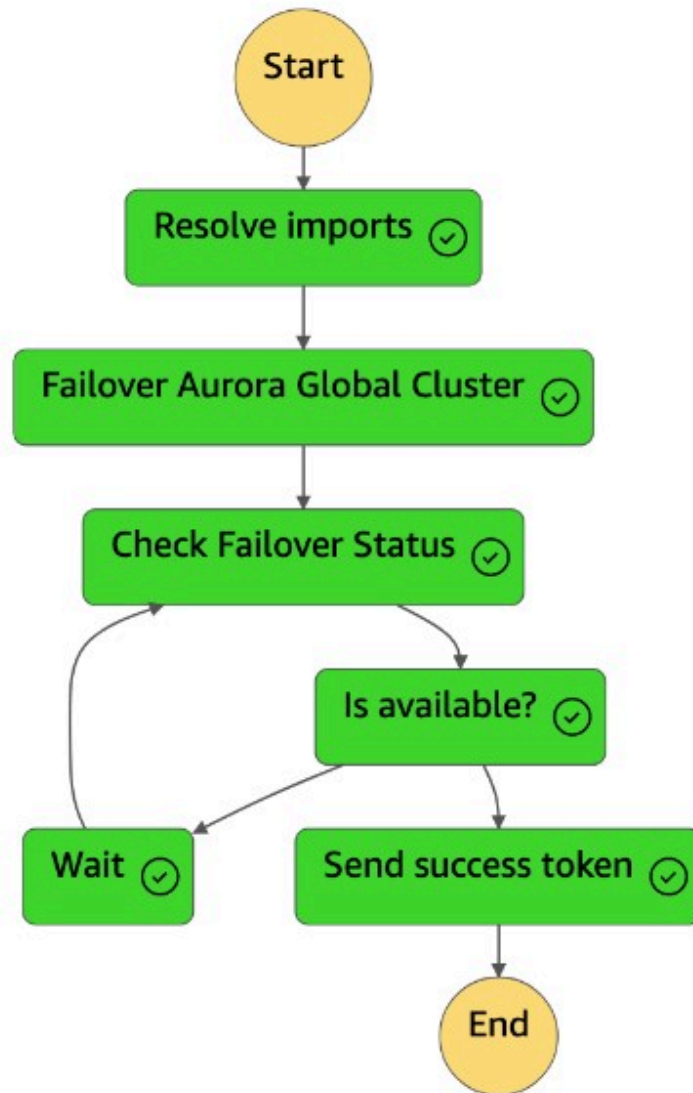
7. dr-orchestrator-stepfunction-planned-Aurora-failover 상태 시스템은 다음 단계를 수행하여 Aurora MySQL 호환 글로벌 데이터베이스 역할을 전환합니다.



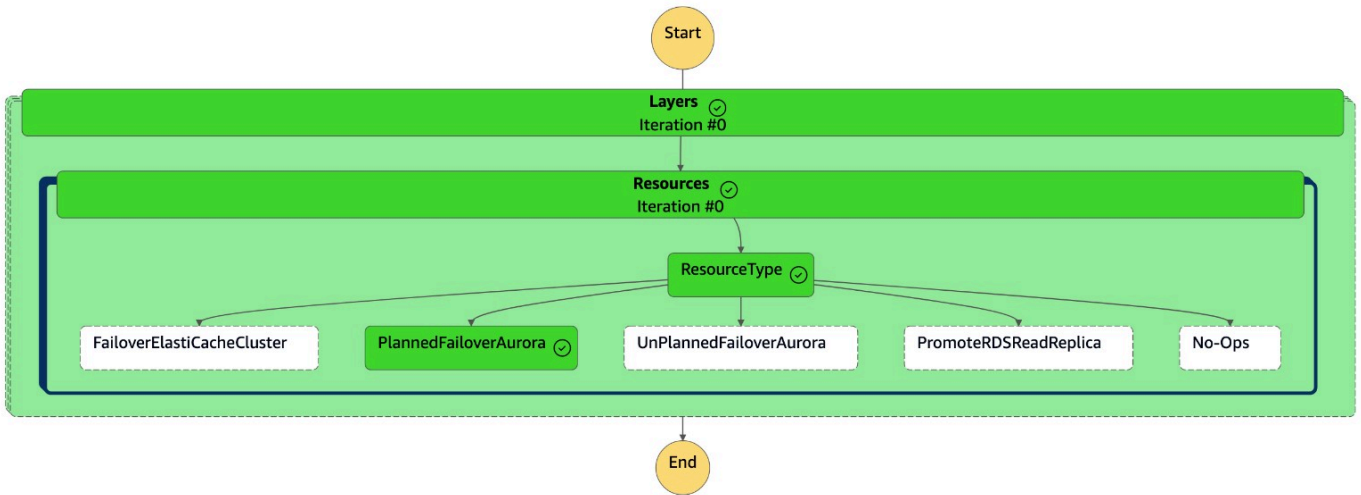
단계	설명	예상 값
가져오기 해결	Lambda 함수는 !Import <variable name> 값을 실제 이름으로 바꿉니다.	"!Import dr-global db-cluster-mysql-global-identifier" 는 로 대체됩니다"dr-global db-cluster-mysql" .

단계	설명	예상 값
장애 조치 Aurora 글로벌 클러스터	Lambda 함수는 failover_global_cluster Boto3 APIs 호출하여 Aurora 글로벌 데이터베이스를 장애 조치합니다.	<pre>{ 'GlobalCluster': { 'GlobalClusterIdentifier': 'dr-global-cluster-mysql', 'GlobalClusterResourceId': 'cluster-cce7f9bec2846db4', 'GlobalClusterArn': 'arn:aws:rds::xxx', 'Status': 'failing-over', } }</pre>
장애 조치 상태 확인	Lambda 함수는 describe_db_clusters Boto3 APIs 호출하여 장애 조치 상태를 확인합니다.	수정, 사용 가능
성공 토큰 전송	Lambda 함수는 send_task_success Boto3 APIs 호출하고 성공 토큰을 DR Orchestrator Failover 상태 시스템으로 다시 보냅니다.	h7XRIcDltd/83p1E0dMccoxlzFhglsdkzpK9mBVKZsp7d9yrT1W

- Amazon RDS 콘솔로 이동합니다. 상태에서 Aurora 글로벌 데이터베이스의 값이 사용 가능에서 전환 또는 수정으로 변경됩니다.
- dr-orchestrator-stepfunction-planned-Aurora-failover 상태 머신이 완료되면 성공 토큰을 dr-orchestrator-stepfunction-FAILOVER 상태 머신으로 다시 보냅니다.



10dr-orchestrator-stepfunction-FAILOVER 상태 머신이 완료되었습니다.



콘솔에서 보조 클러스터(dbcluster-02)의 역할은 이제 기본 클러스터이며 클러스터는 읽기/쓰기 워크로드를 제공할 준비가 되었습니다. 이제 원래 기본 클러스터(dbcluster-01)의 역할이 보조 클러스터로 나열됩니다.

계획되지 않은 수동 장애 조치

드문 경우지만 Aurora 글로벌 데이터베이스의 기본 데이터베이스가 예기치 않게 중단될 수 있습니다. AWS 리전. 이 경우, 기본 Aurora DB 클러스터와 해당 라이터 노드를 사용할 수 없으며 기본 클러스터와 보조 클러스터 간의 복제가 중단됩니다. 가동 중지 시간(RTO)과 데이터 손실(RPO)을 모두 최소화하려면 신속하게 작업하여 리전 간 장애 조치를 수행하고 Aurora 글로벌 데이터베이스를 재구성합니다. 자세한 내용은 [계획되지 않은 중단에서 Amazon Aurora 글로벌 데이터베이스 복구를 참조하세요](#).

계획되지 않은 장애 조치를 수행하려면 Aurora 글로벌 데이터베이스에서 보조 클러스터를 분리해야 합니다. 계획되지 않은 장애 조치를 수행하기 전에 기본 Aurora DB 클러스터에서 애플리케이션 쓰기를 중지합니다. 장애 조치가 성공적으로 완료되면 새 기본 DB 클러스터에 쓰도록 애플리케이션을 재구성합니다. 이 접근 방식은 데이터 손실을 방지하는 데 도움이 됩니다. 또한 장애 조치 프로세스 중에 기본 라이터 노드가 다시 온라인 상태가 되는 경우 데이터 불일치를 방지하는 데 도움이 됩니다.

계획되지 않은 장애 조치를 수행하려면 dr-orchestrator-stepfunction-FAILOVER 상태 시스템을 호출합니다. 이 예제에서 보조 클러스터(db-cluster-02)는 안정 상태의 DR 리전(us-west-2)에 있습니다.

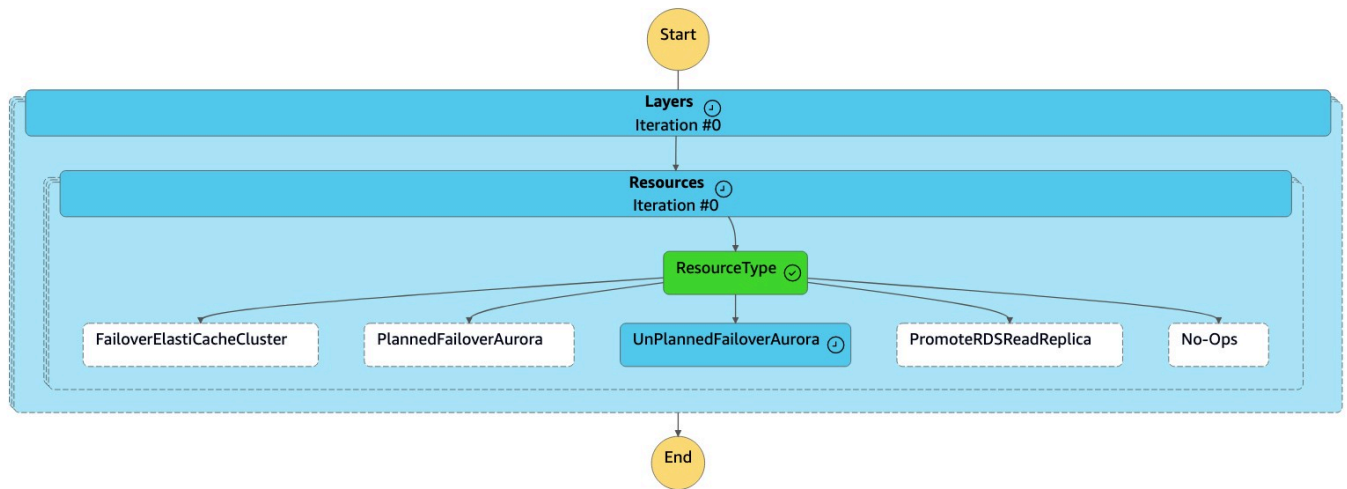
장애 조치를 수행하려면 다음을 수행합니다.

1. 콘솔에 로그인합니다.
2. 리전을 DR 리전(us-west-2)으로 변경합니다.

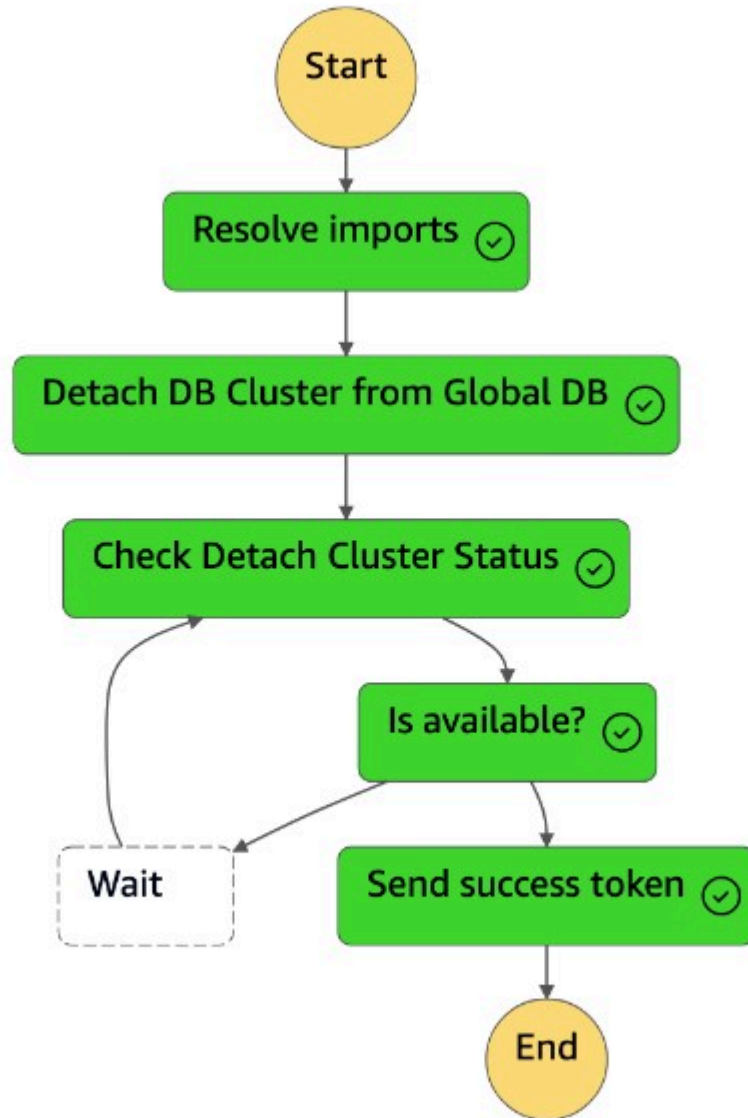
3. 서비스로 이동하여 Step Functions를 선택합니다.
4. dr-orchestrator-stepfunction-FAILOVER 상태 시스템으로 이동합니다.
5. 실행 시작을 선택하고를 UnPlannedFailoverAurora로 사용하여 Input - optional 섹션에 다음 JSON 코드를 입력합니다. resourceType

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

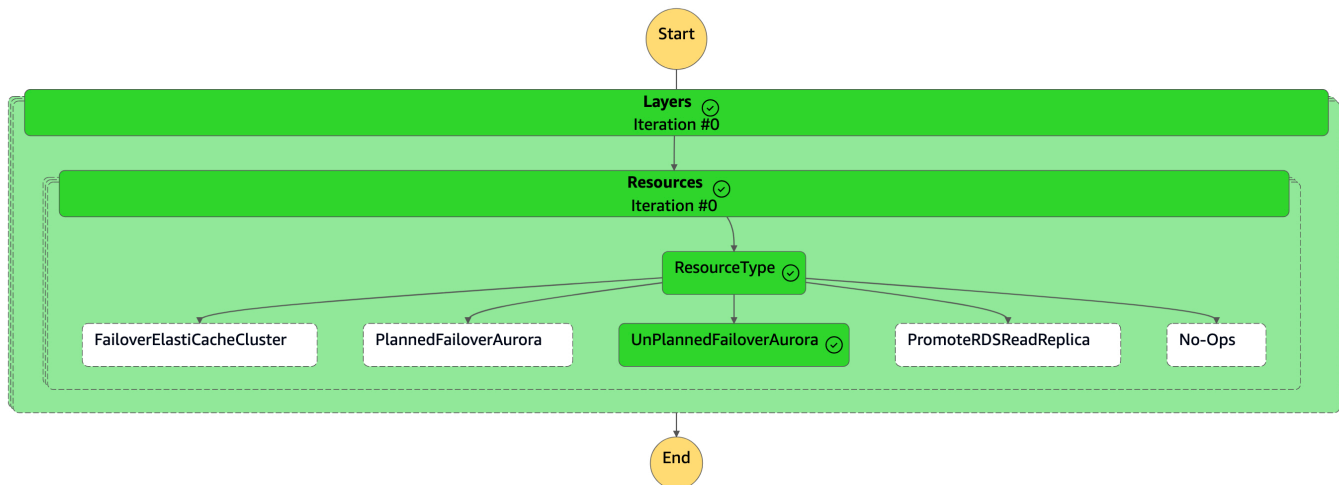
6. dr-orchestrator-stepfunction-FAILOVER 상태 시스템은 리소스 유형을 로 읽고 dr-orchestrator-stepfunction-unplanned-Aurora-failover 상태 시스템에서 작업을 UnPlannedFailoverAuroraMySQL 호출Detach Cluster from Global Database합니다.



7. Detach Cluster from Global Database 작업은 글로벌 데이터베이스에서 보조 클러스터를 분리(제거)합니다.



8. 보조 클러스터(dbcluster-02)는 독립 실행형 클러스터로 승격되며 읽기/쓰기 워크로드를 제공할 수 있습니다.
9. dr-orchestrator-stepfunction-FAILOVER 상태 머신이 완료되었습니다.



10보조 클러스터(dbcluster-02)는 Aurora 글로벌 데이터베이스에서 분리되며 읽기/라이터 워크로드를 처리하는 독립 실행형 클러스터가 됩니다.

11새 클러스터 엔드포인트를 사용하여이 새 독립 실행형 Aurora DB 클러스터로 모든 쓰기 작업을 보내도록 애플리케이션을 재구성합니다.

장애 복구

장애 복구는 재해(또는 예약된 이벤트)가 해결된 후 데이터베이스를 원래(또는 새) 기본 위치로 반환합니다. 계획되지 않은 중단이 해결되면 이전 기본 리전을 Aurora 글로벌 데이터베이스에 다시 추가할 수 있습니다. 먼저 이전 기본 리전에서 기존 DB 클러스터를 삭제하고, 새 기본 리전에서 새 DB 클러스터를 생성한 다음, 계획된 관리형 장애 조치 프로세스를 사용하여 새 클러스터의 역할을 전환해야 합니다.

이는 사용량이 적은 시간이나 주말에 수행할 수 있는 계획된 활동으로 간주될 수 있습니다.

로 생성되었으므로 이전 기본 리전(us-east-1)에서 DR Orchestrator FAILBACK 상태 시스템을 실행DeletionProtection 하기 전에 [Amazon Aurora DB 클러스터를 수동으로 수정](#)하고를 비활성화해야 합니다DeletionProtection.

DR Orchestrator Framework는 dr-orchestrator-stepfunction-FAILBACK 상태 시스템을 사용하여 기존 클러스터를 삭제하고 이전 기본 리전에서 새 클러스터를 생성하는 단계를 자동화합니다.

를 비활성화하려면 다음을 DeletionProtection수행합니다.

1. 콘솔에 로그인합니다.
2. 리전을 이전 기본 리전(us-east-1)으로 변경합니다.

3. Amazon RDS 콘솔로 이동하여 클러스터 이름(dbcluster-01)을 선택하고 수정을 선택합니다.
4. 삭제 방지에서 삭제 방지 활성화 확인란의 선택을 취소하고 계속을 선택합니다.
5. 즉시 적용을 선택한 다음 클러스터 수정을 선택합니다.

DR Orchestrator FAILBACK 상태 시스템은 이전 기본 리전(us-east-1)에서 장애 복구 프로세스 중에 호출됩니다.

장애 복구를 수행하려면 다음을 수행합니다.

1. 콘솔에 로그인합니다.
2. 리전을 이전 기본 리전(us-east-1)으로 변경합니다.
3. 서비스로 이동한 다음 Step Functions를 선택합니다.
4. DR Orchestrator FAILBACK 상태 시스템으로 이동합니다.
5. 실행 시작을 선택하고 Input - optional 섹션에 다음 JSON 코드를 입력합니다.

```
{
  "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",

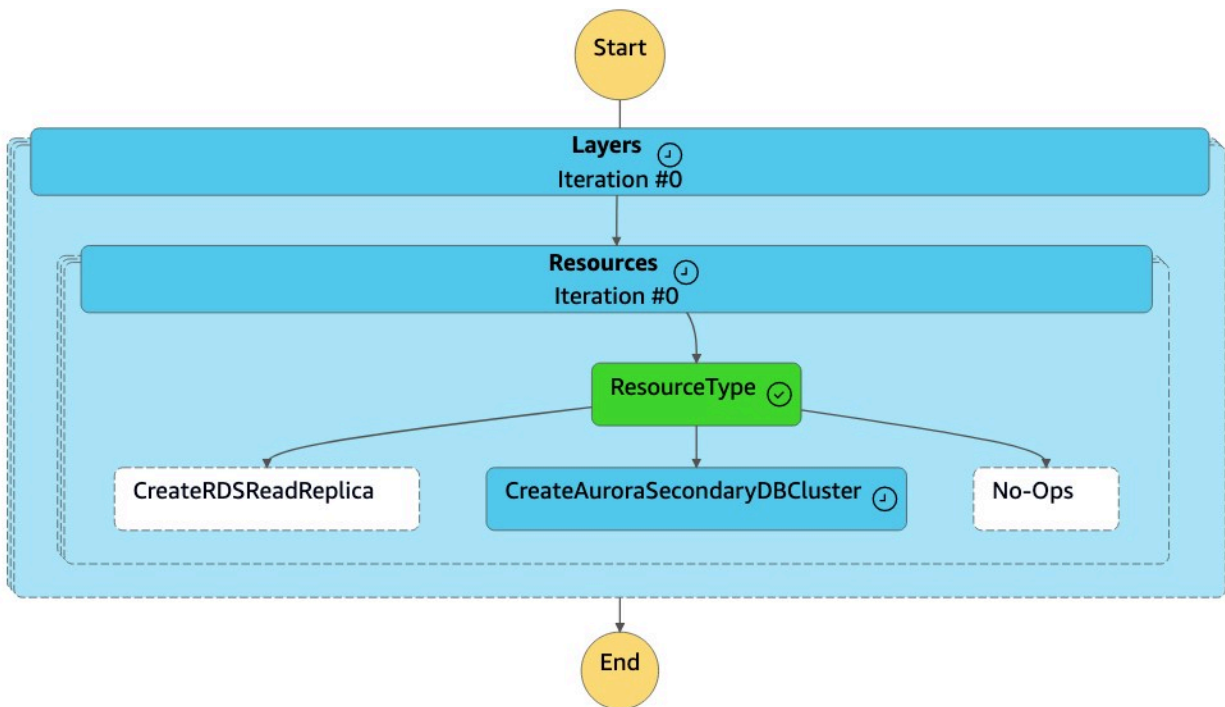
```

```

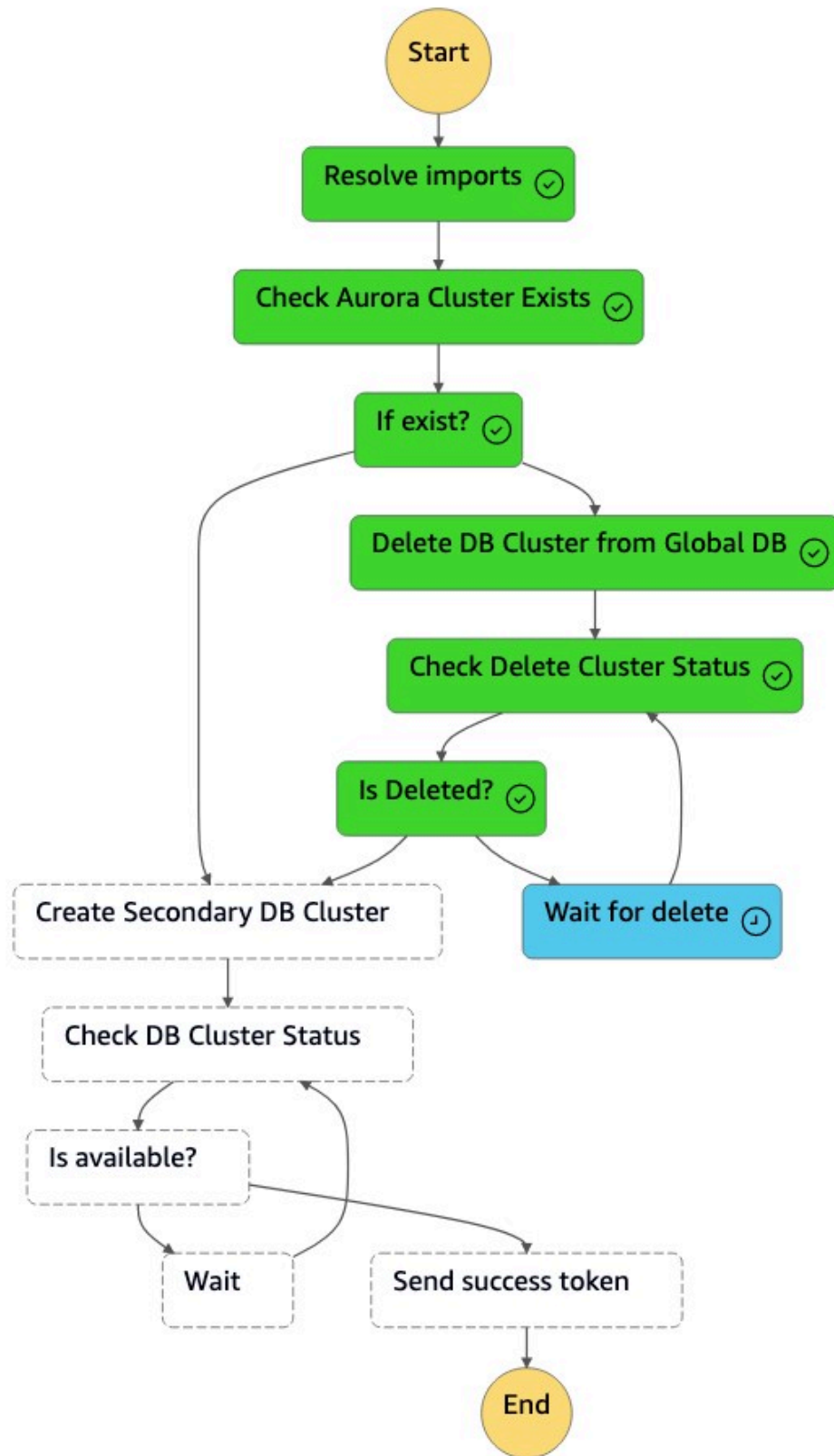
    "EngineVersion": "!Import dr-globaldb-cluster-mysql-engine-version",
    "KmsKeyId": "!Import dr-globaldb-cluster-mysql-KmsKeyId",
    "SourceRegion": "!Import dr-globaldb-cluster-mysql-source-region",
    "ClusterRegion": "!Import dr-globaldb-cluster-mysql-cluster-region",
    "BackupRetentionPeriod": "7",
    "MonitoringInterval": "60",
    "StorageEncrypted": "True",
    "EnableIAMDatabaseAuthentication": "True",
    "DeletionProtection": "True",
    "CopyTagsToSnapshot": "True",
    "AutoMinorVersionUpgrade": "True",
    "MonitoringRoleArn": "!Import rds-mysql-instance-RDSMonitoringRole"
  }
}
]
}
]
}

```

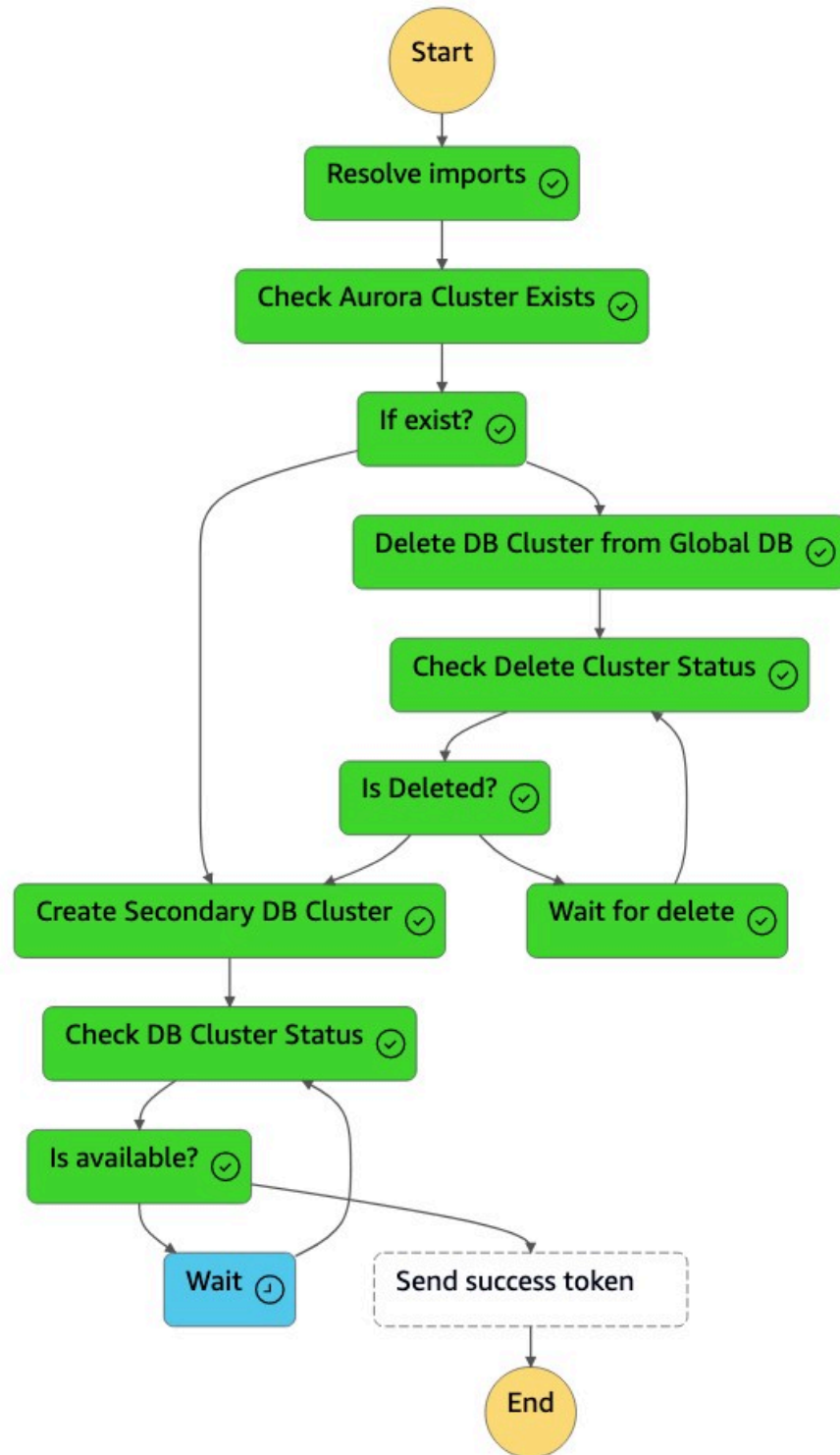
6. DR Orchestrator FAILBACK 상태 시스템은 리소스 유형을 로 읽고 dr-orchestrator-stepfunction-create-Aurora-Secondary-cluster 상태 시스템을 CreateAuroraSecondaryDBCluster호출합니다.



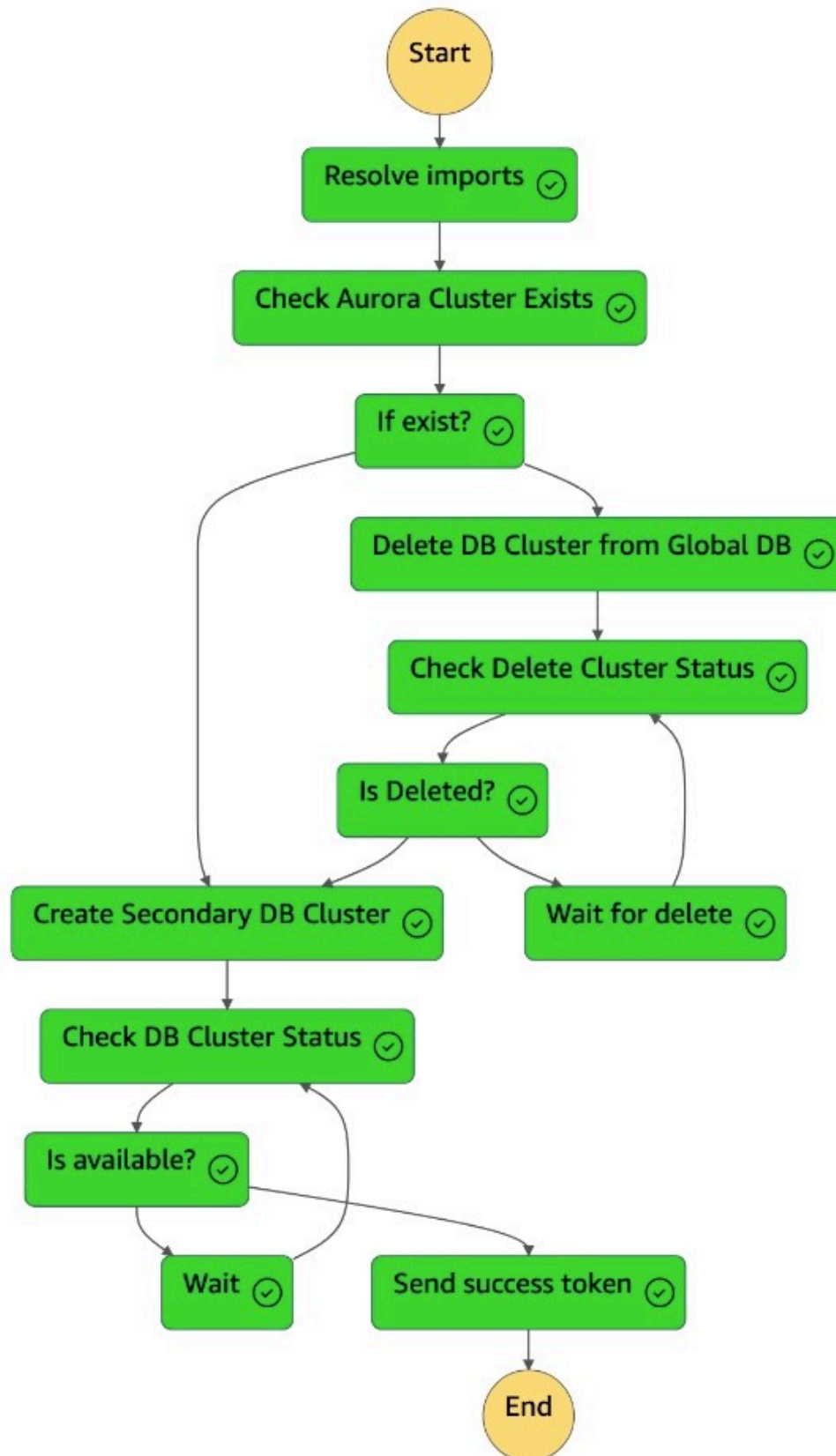
7. `dr-orchestrator-stepfunction-create-Aurora-Secondary-cluster` 상태 시스템은 이전 기본 리전(`dbcluster-01`)에서 기존 클러스터()를 삭제합니다 `us-east-1`.



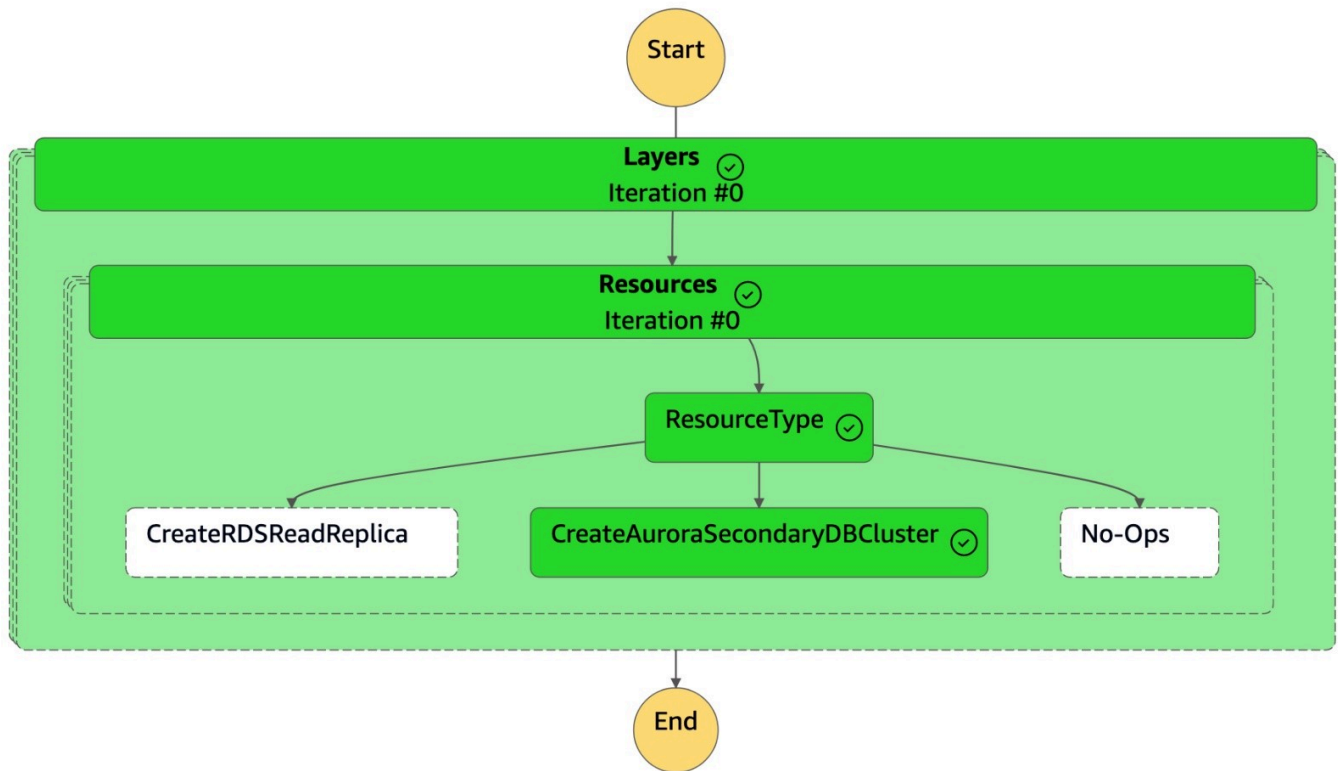
8. 클러스터(dbcluster-01)가 삭제되면 상태 시스템은 DB 인스턴스와 함께 새 클러스터 (dbcluster-01)를 생성하고 Aurora 글로벌 데이터베이스를 보조 클러스터로 조인하여 읽기 전용 워크로드를 제공합니다.



9. 보조 클러스터를 사용할 수 있게 되면 `dr-orchestrator-stepfunction-create-Aurora-Secondary-cluster` 상태 시스템이 완료되고 성공 토큰이 DR Orchestrator Failback 상태 시스템으로 다시 전송됩니다.



10dr-orchestrator-stepfunction-FAILBACK 상태 머신이 완료되었습니다.



11 Amazon RDS 콘솔에서 Aurora 글로벌 데이터베이스를 확인할 수 있습니다.

기본 DB 클러스터를 us-east-1로 재배치하려는 경우 [전환](#) 섹션에 언급된 단계를 따를 수 있습니다.

솔루션 배포

사전 요구 사항을 설정한 후에는 [DR](#) Orchestrator 프레임워크 패턴의 지침을 따르십시오. Amazon 및 RDS Aurora Amazon에 대한 지침이 포함되어 ElastiCache 있습니다. 솔루션을 테스트하려면 사전 요구 사항 및 제한 사항을 검토하고 예제 데이터베이스를 에 배포하십시오. AWS

사전 조건

- 이 솔루션을 적극적으로 [AWS 계정](#) 배포하십시오.
- [Amazon Simple Storage 서비스 \(Amazon S3\)](#) 버킷 및 템플릿에서 사용되는 리소스를 생성할 수 있는 권한. [AWS CloudFormation](#)
- DR Orchestrator FAILBACK 스테이트 머신을 실행하기 전:
 - [RDS 인스턴스를 수동으로 수정하거나 Amazon Aurora DB 클러스터를](#) 수정하고 비활성화합니다. DeletionProtection
 - 근본 원인 분석을 수행하려는 경우 Amazon RDS 인스턴스 또는 Aurora 클러스터를 삭제하기 전에 [DB 스냅샷](#)을 찍으십시오 (RCA).

제한 사항

- 기능 가용성 및 지원은 각 데이터베이스 엔진의 특정 버전과 엔진 전체에 AWS 리전 따라 다릅니다. 지역 간 복제의 기능 및 지역 가용성에 대한 자세한 내용은 지역 간 [읽기 전용](#) 복제본을 참조하십시오.
- 이 솔루션은 읽기 전용 복제본 1개가 있는 기본 데이터베이스 인스턴스를 사용하여 테스트되었습니다. 읽기 전용 복제본을 두 개 이상 사용하려는 경우 프로덕션 환경에 구현하기 전에 솔루션을 철저히 테스트해야 합니다.
- Amazon의 경우 솔루션은 My용 AmazonRDS, [RDSSQLPostgre용 AmazonSQL](#), [RDS RDSMariaDB 용 Amazon과](#) 같은 엔진만 지원합니다.

예제 데이터베이스를 다음에 배포하십시오. AWS

의 데이터베이스를 위한 지역 간 DR 솔루션을 시연하기 위해 Amazon for My AWS, [Amazon RDS Aurora SQL My SQL 호환 에디션 및 ElastiCache Amazon \(OSSRedis\)](#) - 글로벌 데이터스토어를 배포할 수 있습니다. [-database 리포지토리 App-StackREADME 파일에 있는 지침을 따르십시오. aws-cross-region-dr](#) GitHub 이 예제에서 기본 지역은 us-east-1 이고 보조 지역은 us-west-2.

애플리케이션 스택은 AWS CloudFormation [출력](#) 섹션을 사용하여 DR Orchestrator Framework에서 사용하는 매개 변수를 내보내 데이터베이스의 장애 조치 및 장애 복구를 자동화합니다. AWS

FAQ

장애 조치 및 장애 복구를 위한 DR Orchestrator 프레임워크에 대한 질문에 대한 답변을 찾습니다.

이 접근 방식을 사용하여 달성할 수 있는 RPO 및 RTO는 무엇입니까?

Recovery Time Objective(RTO) 및 Recovery Point Objective(RPO)에 대한 자세한 내용은 기반 [데이터베이스에 대한 재해 복구 전략 AWS](#) 전략을 참조하세요.

AWS CloudFormation 내보내기 변수를 사용해야 합니까?

아니요. Amazon Aurora 글로벌 데이터베이스 또는 Amazon RDS DB 인스턴스의 값을 JSON 형식(예: - "RDSInstanceIdentifier": "rds-mysql-instance")으로 직접 전달할 수 있습니다.

DR Orchestrator FAILOVER 워크플로를 사용하여 둘 이상의 AWS 데이터베이스를 장애 조치할 수 있습니까?

예, 입력 파라미터 파일에 둘 이상의 리소스를 전달하여 둘 이상의 AWS 데이터베이스를 장애 조치할 수 있습니다. 다음 코드 예제에서는 Amazon RDS for MySQL 읽기 전용 복제본과 Amazon ElastiCache(Redis OSS) 글로벌 데이터 스토어를 병렬로 장애 조치하는 방법을 보여줍니다.

```
{
  "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 RDS용 DR Orchestrator FAILBACK 상태 시스템을 실행할 때 InvalidParameterCombination 오류를 방지하려면 어떻게 해야 합니까?

오류의 전체 텍스트는 다음과 같습니다.

```

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

```

오류를 방지하려면 DR Orchestrator FAILBACK 상태 시스템을 실행DeletionProtection 하기 전에 비활성화하여 [RDS 인스턴스를 수정](#)합니다.

다음 단계

DR Orchestrator 프레임워크를 사용하여 페일오버 또는 페일백 활동을 시작하기 전에 승인 프로세스를 추가할 수 있습니다. 예를 들어 DR 이벤트 중에 페일오버를 시작하는 동안 이메일을 통해 Amazon [Simple Notification Service \(Amazon SNS\)](#) 알림을 보내는 상태 머신을 추가할 수 있습니다. DR Orchestrator Failover 승인이 승인되는 즉시 장애 조치 활동이 시작됩니다.

애플리케이션 페일오버를 DR Orchestrator Framework와 통합하여 DNS 엔트리 및 포인트 애플리케이션을 데이터베이스 인스턴스 또는 클러스터의 새 엔드포인트로 전환할 수도 있습니다.

[AWS X-Ray](#) 추적을 사용하여 RTO 계산을 위한 페일오버 기간을 확인할 수 있습니다. X-Ray를 기반으로 모니터링 대시보드를 구축할 수 있습니다.

리소스

- [글로벌 데이터 스토어 AWS 리전 를 사용하여 간 복제](#)(Amazon ElastiCache(Redis OSS) 설명서)
- [Using failover in an Amazon Aurora global database](#)(Aurora 설명서)
- [계획된 장애 조치와 계획되지 않은 장애 조치를 위한 Amazon Aurora Global Database 엔드포인트 관리 자동화](#)(블로그 게시물)
- [재해 복구: Terraform을 사용하는 Amazon Aurora 글로벌 데이터베이스에 대한 3가지 장애 조치 시나리오\(2부\)](#)(블로그 게시물)
- [의 데이터베이스에 대한 재해 복구 전략 AWS](#)(AWS 규범적 지침 전략)
- [DR Orchestrator Framework\(권장 가이드 패턴\)를 사용하여 리전 간 장애 조치 및 장애 복구 자동화](#)AWS

문서 이력

아래 표에 이 가이드의 주요 변경 사항이 설명되어 있습니다. 향후 업데이트에 대한 알림을 받으려면 [RSS 피드](#)를 구독하십시오.

변경 사항	설명	날짜
최초 게시	—	2024년 5월 3일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.