개발자 가이드

Amazon MemoryDB

Amazon MemoryDB: 개발자 가이드

Table of Contents

MemoryDB란?

MemoryDB는 초고속 성능을 제공하는, 내구성이 뛰어난 인 메모리 데이터베이스 서비스입니다. 마이크로서비스 아키텍처를 사용하는 최신 애플리케이션을 위해 특별히 제작되었습니다.

Amazon MemoryDB는 널리 사용되는 오픈 소스 데이터 저장소인 Valkey 및 Redis OSS와 호환되므로 이미 사용하고 있는 것과 동일한 유연하고 친숙한 데이터 구조, API 및 명령을 사용하여 애플리케이션을 빠르게 구축할 수 있습니다. MemoryDB를 사용하면 모든 데이터가 메모리에 저장되므로 마이크로초의 읽기 및 한 자릿수 밀리초의 쓰기 지연 시간과 높은 처리량을 달성할 수 있습니다. 또한 MemoryDB는 다중 AZ 트랜잭션 로그를 사용하여 여러 가용 영역(AZ)에 데이터를 안정적으로 저장하므로 장애 조치, 데이터베이스 복구 및 노드 재시작이 빠릅니다.

메모리 내 성능과 다중 AZ 내구성을 모두 제공하는 MemoryDB는 마이크로서비스 애플리케이션을 위한 고성능 기본 데이터베이스로 사용할 수 있으므로 캐시와 내구성이 뛰어난 데이터베이스를 별도로 관리할 필요가 없습니다.

주제

MemoryDB 기능

MemoryDB는 초고속 성능을 제공하는, 내구성이 뛰어난 인 메모리 데이터베이스 서비스입니다. MemoryDB의 기능은 다음과 같습니다.

- 프라이머리 노드의 강력한 일관성과 복제본 노드의 최종 일관성 보장 자세한 내용은 일관성 섹션을 참조하세요.
- 클러스터당 최대 1억 6천만 TPS의 마이크로초 읽기 및 한 자릿수 밀리초의 쓰기 지연 시간을 제공합니다.

- 유연하고 친숙한 Valkey 및 Redis OSS 데이터 구조 및 API 거의 수정하지 않고도 새 애플리케이션을 쉽게 구축하거나 기존 Valkey 기반 및 Redis OSS 기반 애플리케이션을 마이그레이션할 수 있습니다.
- 빠른 데이터베이스 복구 및 재시작을 제공하는 다중 AZ 트랜잭션 로그를 사용한 데이터 내구성.
- 자동 장애 조치, 노드 장애 감지 및 복구 기능을 갖춘 다중 AZ 가용성
- 노드를 추가 및 제거하여 수평적으로 규모를 조정하거나 더 크거나 작은 노드 유형으로 이동하여 수직적으로 규모를 조정할 수 있습니다. 샤드를 추가하여 쓰기 처리량을 확장하고 복제본을 추가하여 읽기 처리량을 확장할 수 있습니다.
- 프라이머리 노드의 읽기 후 쓰기 일관성과 복제본 노드의 최종 일관성이 보장됩니다.
- MemoryDB는 전송 중 데이터 암호화, 저장 중 데이터 암호화 및 액세스 제어 목록(ACL)을 사용하여 사용자 인증을(를) 통한 사용자 인증을 지원합니다.
- Amazon S3의 자동 스냅샷으로 최대 35일간 보존할 수 있습니다.
- 클러스터당 최대 500개의 노드와 100TB 이상의 스토리지(샤드당 복제본 1개)를 지원합니다.
- TLS를 사용한 전송 중 데이터 암호화 및 AWS KMS 키를 사용한 저장 중 데이터 암호화.
- Valkey 및 Redis OSS를 통한 사용자 인증 및 권한 부여 액세스 제어 목록(ACL)을 사용하여 사용자 인증.
- AWS Graviton2 인스턴스 유형을 지원합니다.
- 모니터링, 보안 및 알림을 위해 CloudWatch, Amazon VPC, CloudTrail, Amazon SNS와 같은 다른 AWS 서비스와의 통합
- 완전 관리형 소프트웨어 패치 및 업그레이드.
- 관리 API를 위한 AWS Identity and Access Management(IAM) 통합 및 태그 기반 액세스 제어입니다.

MemoryDB 코어 구성 요소

다음은 MemoryDB용 배포의 주요 구성 요소를 개괄적으로 설명한 것입니다.

주제

- 액세스 통제 목록
- 사용자

클러스터

클러스터는 단일 데이터 세트를 제공하는 하나 이상의 노드 모음입니다. MemoryDB 데이터 세트는 샤드별로 분할되어 있으며, 각 샤드에는 읽기/쓰기 프라이머리 노드 및 최대 5개의 선택적 복제본 노드가 있습니다. 프라이머리 노드는 읽기 및 쓰기 요청을 처리하는 반면 복제본은 읽기 요청만 처리합니다. 프라이머리 노드는 복제본 노드로 장애 조치하여 해당 복제본을 해당 샤드의 새 프라이머리 노드로 승격시킬 수 있습니다. MemoryDB는 Valkey 또는 Redis OSS를 데이터베이스 엔진으로 실행하며, 클러스터를 생성할 때 클러스터의 엔진 버전을 지정합니다. AWS CLI, MemoryDB API 또는 AWS Management 콘솔를 사용하여 클러스터를 수정할 수 있습니다.

각 MemoryDB 클러스터는 Valkey 또는 Redis OSS 엔진 버전을 실행합니다. 각 엔진 버전에는 지원되는 고유한 기능이 있습니다. 또한 엔진 버전마다 관리하는 클러스터의 동작을 제어하는 파라미터 그룹에 파라미터 집합이 있습니다.

클러스터의 계산 및 메모리 용량은 해당 노드 유형에 의해 결정됩니다. 사용자의 요구 사항에 가장 잘 맞는 노드 유형을 선택할 수 있습니다. 시간이 지나면서 요구 사항이 바뀌면 노드 유형을 변경할 수 있습니다. 자세한 내용은 지원되는 노드 유형을(를) 참조하세요.

> **Note**
>
> MemoryDB 노드 유형에 대한 요금 정보는 MemoryDB 요금을 참조하세요.

Amazon Virtual Private Cloud(VPC) 서비스를 사용해 Virtual Private Cloud(VPC)에서 클러스터를 실행합니다. VPC를 사용하면 가상 네트워킹 환경을 완벽하게 제어할 수 있습니다. 자기만의 IP 주소 범위를 선택하고, 서브넷을 생성하고, 라우팅 및 액세스 제어 목록을 구성할 수 있습니다. MemoryDB는 스냅샷, 소프트웨어 패치, 자동 장애 감지 및 복구를 관리합니다. VPC에서 클러스터를 실행하는 데는 추가 비용이 들지 않습니다. MemoryDB와 Amazon VPC를 함께 사용하는 방법에 대한 자세한 내용은 MemoryDB 및 Amazon VPC 단원을 참조하세요.

여러 MemoryDB 작업은 클러스터에서 다음 사항을 대상으로 합니다.

- 클러스터 생성
- 클러스터 수정
- 클러스터의 스냅샷 생성

- 클러스터 삭제
- 클러스터의 요소 보기
- 비용 할당 태그를 클러스터에 추가 및 클러스터에서 삭제

자세한 내용은 다음 관련 항목을 참조하세요.

- 클러스터 관리'and'노드 관리

  클러스터, 노드 및 관련 작업에 대한 정보입니다.
- MemoryDB의 복원력

  클러스터의 내결함성 향상에 관한 정보입니다.

노드

노드는 MemoryDB 배포의 가장 작은 구성 요소이며 Amazon EC2 인스턴스를 사용하여 실행됩니다. 각 노드는 클러스터를 생성할 때 선택한 엔진 버전을 실행합니다. 노드는 클러스터에 속하는 샤드에 속합니다.

각 노드는 클러스터를 생성할 때 선택한 버전으로 엔진 인스턴스를 실행합니다. 필요한 경우, 클러스터의 노드를 다른 유형으로 스케일 업하거나 스케일 다운할 수 있습니다. 자세한 내용은 스케일링 섹션을 참조하세요.

클러스터 내 모든 노드는 동일한 노드 유형입니다. 여러 유형의 노드가 지원되며 메모리 양이 각각 다릅니다. 지원되는 노드 유형의 전체 목록은 지원되는 노드 유형 단원을 참조하세요.

노드에 대한 자세한 내용은 노드 관리 섹션을 참조하세요.

샤드

샤드는 1~6개의 노드를 그룹화한 것으로, 하나는 기본 쓰기 노드로, 나머지 5개는 읽기 복제본으로 사용됩니다. MemoryDB 클러스터에는 항상 샤드가 하나 이상 있습니다.

MemoryDB 클러스터는 샤드에서 데이터가 분할된 최대 500개의 샤드를 포함할 수 있습니다. 예를 들어 83개 샤드(샤드당 기본 1개와 복제본 5개)에서 500개 샤드(기본 1개와 복제본 없음) 범위의 500개 노드 클러스터를 구성하도록 선택할 수 있습니다. 증가를 수용할 수 있는 IP 주소가 충분한지 확인해야 합니다. 서브넷 그룹에 있는 서브넷의 CIDR 범위가 너무 작거나 서브넷을 샤드로 분할하여 다른 클러스터에서 과도하게 사용되는 것과 같은 일반적인 함정에 유의합니다.

다중 노드 샤드는 읽기/쓰기 기본 노드 하나와 1~5개의 복제본 노드를 통해 복제를 구현합니다. 자세한 내용은 MemoryDB 복제 이해 섹션을 참조하세요.

샤드에 대한 자세한 내용은 샤드 작업 섹션을 참조하세요.

파라미터 그룹

파라미터 그룹을 사용하면 클러스터의 엔진에 대한 런타임 설정을 쉽게 관리할 수 있습니다. 메모리 사용량, 항목 크기 등을 제어하는 데 여러 가지 파라미터가 사용됩니다. MemoryDB 파라미터 그룹은 클러스터에 적용할 수 있는 명명된 엔진별 파라미터 모음입니다. 이를 통해 해당 클러스터에 있는 모든 노드가 정확히 동일한 방법으로 구성됩니다.

MemoryDB 파라미터 그룹에 대한 자세한 내용은 파라미터 그룹을 사용해 엔진 파라미터 구성 단원을 참조하세요.

서브넷 그룹

서브넷 그룹은 Amazon Virtual Private Cloud(VPC) 환경에서 실행 중인 클러스터에 대해 지정할 수 있는 서브넷(일반적으로 프라이빗 서브넷) 모음입니다.

Amazon VPC에서 클러스터를 생성하는 경우 서브넷 그룹을 지정하거나 제공된 기본 서브넷 그룹을 사용할 수 있습니다. MemoryDB는 해당 서브넷 그룹을 사용하여 노드에 연결된 서브넷 내의 서브넷 및 IP 주소를 선택합니다.

MemoryDB 서브넷 그룹에 대한 자세한 내용은 서브넷 및 서브넷 그룹 섹션을 참조하세요.

액세스 통제 목록

액세스 제어 목록은 한 명 이상의 사용자 모음입니다. 액세스 문자열은 Redis ACL 규칙에 따라 Valkey 또는 Redis OSS 명령 및 데이터에 대한 사용자 액세스를 승인합니다.

MemoryDB 액세스 제어 목록에 대한 자세한 내용은 액세스 제어 목록(ACL)을 사용하여 사용자 인증을(를) 참조하세요.

사용자

사용자는 사용자 이름과 암호를 가지고 있으며, 이 사용자 이름은 MemoryDB 클러스터의 데이터에 액세스하고 명령을 실행하는 데 사용됩니다. 사용자는 액세스 제어 목록(ACL) 의 구성원이며, 액세스 제어 목록(ACL) 을 사용하여 MemoryDB 클러스터에서 해당 사용자의 권한을 확인할 수 있습니다. 자세한 내용은 액세스 제어 목록(ACL)을 사용하여 사용자 인증 섹션을 참조하세요.

관련 서비스

ElastiCache

MemoryDB와 ElastiCache 중 어느 것을 사용할지 결정할 때는 다음과 같은 점을 비교하세요.

- MemoryDB는 초고속 기본 데이터베이스가 필요한 워크로드에 적합한, 내구성이 뛰어난 인 메모리 데이터베이스입니다. 워크로드에 초고속 성능(마이크로초 단위 읽기 및 10밀리초의 쓰기 지연 시간)을 제공하는 내구성이 뛰어난 데이터베이스가 필요한 경우, MemoryDB 사용을 고려해야 합니다. Valkey 또는 Redis OSS 데이터 구조 및 API와 함께 내구성이 뛰어난 기본 데이터베이스를 사용하여 애플리케이션을 구축하려는 경우, MemoryDB가 사용 사례에 적합할 수 있습니다. 마지막으로, 내구성과 성능을 위해 데이터베이스 사용을 캐시로 대체하여 애플리케이션 아키텍처를 단순화하고 비용을 절감하려면 MemoryDB 사용을 고려해야 합니다.
- ElastiCache는 일반적으로 Valkey 및 Redis OSS를 사용하여 다른 데이터베이스 및 데이터 저장소의 데이터를 캐시하는 데 사용되는 서비스입니다. 워크로드를 캐싱하여 기존 기본 데이터베이스 또는 데이터 저장소(마이크로초 단위 읽기 및 쓰기 성능)로 데이터 액세스를 가속화하려면 ElastiCache를 고려해야 합니다. Valkey 또는 Redis OSS 데이터 구조 및 API를 사용하여 기본 데이터베이스 또는 데이터 저장소에 저장된 데이터에 액세스하려는 사용 사례에도 ElastiCache를 고려해야 합니다.

리전 및 가용 영역 선택

AWS 클라우드 컴퓨팅 리소스는 가용성이 매우 높은 데이터 설비에 있습니다. 추가 확장성 및 신뢰성을 제공하기 위해 이러한 데이터 센터 시설은 여러 물리적 위치에 배치됩니다. 이러한 위치는 리전 및 가용 영역으로 분류됩니다.

AWS 리전은 크고 광범위하게 별도의 지리적 위치에 분산되어 있습니다. 가용 영역은 다른 가용 영역에서 발생한 장애로부터 격리할 수 있도록 설계된 AWS 리전 내 개별적인 위치입니다. 가용 영역은 같은 AWS 리전에 있는 다른 가용 영역에 대해 저렴하고 지연 시간이 짧은 네트워크 연결을 제공합니다.

> **Important**
>
> 각 리전은 완전히 독립적입니다. 시작하는 모든 MemoryDB 활동(예: 클러스터 생성)은 현재 기본 리전에서만 실행됩니다.

특정 리전에서 클러스터를 생성하거나 사용하려면 해당하는 리전 서비스 엔드포인트를 사용하세요. 서비스 엔드포인트는 MemoryDB Multi-Region 섹션을 참조하세요.

MemoryDB Multi-Region을 사용하면 가용성과 복원력을 모두 향상시키는 동시에 Multi-Region 애플리케이션의 로컬 읽기 및 쓰기 지연 시간이 짧은 이점을 누릴 수 있습니다. MemoryDB Multi-Region 작업에 대한 자세한 내용은 지원되는 리전 및 엔드포인트 섹션을 참조하세요.

노드 찾기

하나 이상의 복제본이 있는 클러스터는 AZ에서 분산되어 있어야 합니다. 단일 AZ 내에서 모든 항목을 찾을 수 있는 유일한 방법은 단일 노드 샤드로 구성된 클러스터를 사용하는 것입니다.

서로 다른 AZ에 노드를 배치함으로써 MemoryDB는 하나의 AZ에서 정전과 같은 장애가 발생할 경우, 가용성이 손실될 가능성을 제거합니다.

- MemoryDB 클러스터 생성
- MemoryDB 클러스터 수정

지원되는 리전 및 엔드포인트

MemoryDB는 여러 AWS 리전에서 사용할 수 있습니다. 따라서 요구 사항에 적합한 위치에서 MemoryDB 클러스터를 시작할 수 있습니다. 예를 들어, 고객과 가장 가까운 AWS 리전 또는 특정 법적 요구 사항을 준수하는 AWS 리전에서 시작할 수 있습니다. 또한 MemoryDB가 새로운 AWS 리전으로 가용성을 확대함에 따라 MemoryDB는 새로운 리전에 당시의 최신 `MAJOR.MINOR` 버전 두 개를 지원합니다. MemoryDB 버전에 대한 자세한 내용은 엔진 버전 섹션을 참조하세요.

기본적으로 AWS SDK, AWS CLI, MemoryDB API 및 MemoryDB 콘솔은 미국 동부 (버지니아 북부) 리전을 참조합니다. MemoryDB를 새 리전에서 사용할 수 있게 됨에 따라 이러한 리전의 새 엔드포인트 또한 HTTP 요청, AWS SDK, AWS CLI 및 콘솔에서 사용할 수 있습니다.

각 리전은 다른 리전에서 완전히 격리되도록 설계되었습니다. 각 리전 안에는 가용 영역(AZ)가 여러 개 있습니다. 서로 다른 가용 영역에서 노드를 시작하면 가능한 최고 수준의 내결함성을 갖출 수 있습니다. 리전 및 가용 영역에 대한 자세한 내용은 이 주제의 맨 위에 있는 리전 및 가용 영역 선택 섹션을 참조하세요.

MemoryDB가 지원되는 리전

리전 이름/리전	Endpoint	프로토콜	
미국 동부(오하이오) 리전 `us-east-2`	`memory-db.us-east-2.amazonaws.com`	HTTPS	
미국 동부(버지니아 북부) 리전 `us-east-1`	`memory-db.us-east-1.amazonaws.com`	HTTPS	
미국 서부(캘리포니아 북부) 리전 `us-west-1`	`memory-db.us-west-1.amazonaws.com`	HTTPS	
미국 서부(오리건) 리전 `us-west-2`	`memory-db.us-west-2.amazonaws.com`	HTTPS	

리전 이름/리전	Endpoint	프로토콜
캐나다(중부) 리전 ca-central-1	memory-db.ca-central-1.amazonaws.com	HTTPS
아시아 태평양(홍콩) 리전 ap-east-1	memory-db.ap-eastl-1.amazonaws.com	HTTPS
아시아 태평양(뭄바이) 리전 ap-south-1	memory-db.ap-south-1.amazonaws.com	HTTPS
아시아 태평양(도쿄) 리전 ap-northeast-1	memory-db.ap-northeast-1.amazonaws.com	HTTPS
아시아 태평양(서울) 리전 ap-northeast-2	memory-db.ap-northeast-2.amazonaws.com	HTTPS
아시아 태평양(싱가포르) 리전 ap-southeast-1	memory-db.ap-southeast-1.amazonaws.com	HTTPS
아시아 태평양(시드니) 리전 ap-southeast-2	memory-db.ap-southeast-2.amazonaws.com	HTTPS
유럽(프랑크푸르트) 리전 eu-central-1	memory-db.eu-central-1.amazonaws.com	HTTPS

리전 이름/리전	Endpoint	프로토콜	
유럽(아일랜드) 리전 eu-west-1	memory-db.eu-west-1.amazonaws.com	HTTPS	
유럽(런던) 리전 eu-west-2	memory-db.eu-west-2.amazonaws.com	HTTPS	
EU(파리) 리전 eu-west-3	memory-db.eu-west-3.amazonaws.com	HTTPS	
유럽(스톡홀름) 리전 eu-north-1	memory-db.eu-north-1.amazonaws.com	HTTPS	
유럽(밀라노) 리전 eu-south-1	memory-db.eu-south-1.amazonaws.com	HTTPS	
유럽(스페인) 리전 eu-south-2	memory-db.eu-south-2.amazonaws.com	HTTPS	
남아메리카(상파울루) 리전 sa-east-1	memory-db.sa-east-1.amazonaws.com	HTTPS	
중국(베이징) 리전 cn-north-1	memory-db.cn-north-1.amazonaws.com.cn	HTTPS	
중국(닝샤) 리전 cn-northwest-1	memory-db.cn-northwest-1.amazonaws.com.cn	HTTPS	

리전별 AWS 제품 및 서비스 표는 리전별 제품 및 서비스에서 참조하세요.

리전 내에서 지원되는 가용 영역 표는 서브넷 및 서브넷 그룹을(를) 참조하세요.

MemoryDB에 액세스

각 MemoryDB 클러스터 엔드포인트에는 주소와 포트가 있습니다. 이 클러스터 엔드포인트는 클라이언트가 클러스터의 각 노드에 대한 특정 역할, IP 주소 및 슬롯을 검색할 수 있도록 Valkey 및 Redis OSS 클러스터 프로토콜을 지원합니다. 프라이머리 노드에 장애가 발생하고 복제본이 대신 승격되면 Valkey 또는 Redis OSS 클러스터 프로토콜을 사용해 클러스터 엔드포인트에 연결하여 새 프라이머리 노드를 검색할 수 있습니다.

cluster nodes 또는 cluster slots 명령을 사용하여 노드 엔드포인트를 검색하려면 클러스터 엔드포인트에 연결해야 합니다. 키에 적합한 노드를 검색한 후 해당 노드에 직접 연결하여 읽기/쓰기 요청을 수행할 수 있습니다. Valkey 또는 Redis OSS 클라이언트는 클러스터 엔드포인트를 사용하여 올바른 노드에 자동으로 연결할 수 있습니다.

클러스터의 특정 노드 문제를 해결하기 위해 노드별 엔드포인트를 사용할 수도 있지만 일반적인 사용에는 필요하지 않습니다.

클러스터의 엔드포인트를 찾으려면 다음을 참조하세요.

- MemoryDB 클러스터(CLI AWS)의 엔드포인트 찾기
- MemoryDB 클러스터의 엔드포인트 찾기(MemoryDB API)

노드나 클러스터에 연결하는 방법은 redis-cli를 사용하여 MemoryDB 노드에 연결을(를) 참조하세요.

MemoryDB 보안

MemoryDB 보안은 다음과 같이 세 가지 수준에서 관리됩니다.

- MemoryDB 클러스터 및 노드에서 관리 작업을 수행할 수 있는 사용자를 제어하려면 AWS Identity and Access Management(IAM)을 사용합니다. IAM 보안 인증 정보를 사용하여 AWS에 연결할 때는 작업에 필요한 권한을 부여할 수 있는 IAM 정책이 AWS 계정에 반드시 필요합니다. 자세한 내용은 MemoryDB의 ID 및 액세스 관리 섹션을 참조하세요.
- 클러스터에 대한 액세스 수준을 제어하려면 지정된 권한을 가진 사용자를 생성하고 액세스 제어 목록(ACL)에 할당합니다. 그러면 ACL이 하나 이상의 클러스터와 차례로 연결됩니다. 자세한 내용은 액세스 제어 목록(ACL)을 사용하여 사용자 인증 섹션을 참조하세요.

- Aurora DB 클러스터는 Amazon VPC 서비스를 기반으로 Virtual Private Cloud(VPC)에 생성해야 합니다. VPC에 있는 MemoryDB 클러스터 노드의 엔드포인트 및 포트에 연결할 수 있는 디바이스와 Amazon EC2 인스턴스를 제어하려면 VPC 보안 그룹을 사용합니다. 전송 계층 보안(TLS)/Secure Sockets Layer(SSL)를 사용하여 이러한 엔드포인트 및 포트 연결을 만들 수 있습니다. 그 밖에도 기업의 방화벽 규칙을 통해 기업에서 이용하는 디바이스의 MemoryDB 클러스터 연결 여부를 제어하는 것도 가능합니다. VPC에 대한 자세한 내용은 MemoryDB 및 Amazon VPC 단원을 참조하세요.

구성 보안에 대한 자세한 내용은 MemoryDB의 보안 단원을 참조하세요.

MemoryDB 시작하기

이 연습에서는 MemoryDB 관리 콘솔을 사용하여 MemoryDB 클러스터를 만들고, 액세스 권한을 부여하고, 연결하고, 마지막으로 삭제하는 단계를 안내합니다.

> **Note**
>
> 이 연습에서는 클러스터를 생성할 때 간편 생성 옵션을 사용하고 MemoryDB의 기능을 더 자세히 살펴본 후에 다른 두 가지 옵션을 사용하는 것이 좋습니다.

주제

1단계: 설정

다음 주제에서는 MemoryDB 사용을 시작하기 위해 수행해야 하는 1회성 작업에 관해 설명합니다.

에 가입AWS 계정

이 없는 경우 다음 단계를 AWS 계정완료하여 생성합니다.

에 가입하려면AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 엽니다.
2. 온라인 지침을 따릅니다.

   등록 절차 중 전화 또는 텍스트 메시지를 받고 전화 키패드로 확인 코드를 입력하는 과정이 있습니다.

   에 가입하면AWS 계정 AWS 계정 루트 사용자이 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스에 액세스할 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스

권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS는 가입 프로세스가 완료된 후 확인 이메일을 보냅니다. 언제든지 https://aws.amazon.com/으로 이동하고 내 계정을 선택하여 현재 계정 활동을 확인하고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

에 가입한 후 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 AWS 계정보호AWS IAM Identity Center, AWS 계정 루트 사용자활성화 및 생성합니다.

보안AWS 계정 루트 사용자

1. 루트 사용자를 선택하고 AWS 계정이메일 주소를 입력하여 계정 소유자AWS Management 콘솔로에 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

   루트 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 사용 설명서의 루트 사용자로 로그인을 참조하세요.
2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

   지침은 IAM 사용 설명서의 AWS 계정루트 사용자(콘솔)에 대한 가상 MFA 디바이스 활성화를 참조하세요.

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

   지침은 AWS IAM Identity Center 사용자 안내서의 AWS IAM Identity Center 설정을 참조하세요.
2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

   를 자격 증명 소스IAM Identity Center 디렉터리로 사용하는 방법에 대한 자습서는 사용 AWS IAM Identity Center설명서의 기본값으로 사용자 액세스 구성을IAM Identity Center 디렉터리 참조하세요.

관리 액세스 권한이 있는 사용자로 로그인

- IAM IDentity Center 사용자로 로그인하려면 IAM Identity Center 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.

IAM Identity Center 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인사용 설명서의 AWS액세스 포털에 로그인을 참조하세요.

추가 사용자에게 액세스 권한 할당

1. IAM Identity Center에서 최소 권한 적용 모범 사례를 따르는 권한 세트를 생성합니다.

   지침은 Create a permission set에 대한 지침은 AWS IAM Identity Center사용 설명서에서 참조하세요.

2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 Add groups를 참조하세요.

프로그래밍 방식 액세스 권한 부여

사용자는 AWS외부에서와 상호 작용하려는 경우 프로그래밍 방식의 액세스가 필요합니다AWS Management 콘솔. 프로그래밍 방식 액세스를 부여하는 방법은에 액세스하는 사용자 유형에 따라 다릅니다AWS.

사용자에게 프로그래밍 방식 액세스 권한을 부여하려면 다음 옵션 중 하나를 선택합니다.

프로그래밍 방식 액세스가 필요한 사용자	목적	방법
IAM	(권장) 콘솔 자격 증명을 임시 자격 증명으로 사용하여 AWS CLI,AWSSDKs 또는 AWSAPIs.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • 자세한 AWS CLI내용은 AWS Command Line Interface사용 설명서의 AWS로컬 개발을 위한 로그인을 참조하세요. • AWSSDKs 경우 SDK 및 도구 참조 안내서의 AWS로컬 개발을 위한 로그인을 참조하세요. AWSSDKs

프로그래밍 방식 액세스가 필요한 사용자	목적	방법
작업 인력 ID (IAM Identity Center에서 관리되는 사용자)	임시 자격 증명을 사용하여 AWS CLI,AWSSDKs 또는 AWSAPIs.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • 자세한 AWS CLI내용은 AWS Command Line Interface사용 설명서의 AWS CLI를 사용하도록 구성을AWS IAM Identity Center 참조하세요. • AWSSDKs, 도구 및 AWSAPIs 경우 SDK 및 도구 참조 안내서의 IAM Identity Center 인증을 참조하세요. AWSSDKs
IAM	임시 자격 증명을 사용하여 AWS CLI,AWSSDKs 또는 AWSAPIs.	IAM 사용 설명서의 AWS리소스에서 임시 자격 증명 사용의 지침을 따릅니다.

프로그래밍 방식 액세스가 필요한 사용자	목적	방법
IAM	(권장되지 않음) 장기 자격 증명을 사용하여 AWS CLI,AWSSDKs 또는 AWSAPIs.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • 자세한 AWS CLI내용은 AWS Command Line Interface사용 설명서의 IAM 사용자 자격 증명을 사용하여 인증을 참조하세요. • AWSSDKs 및 도구의 경우 SDK 및 도구 참조 안내서의 장기 자격 증명을 사용하여 인증을 참조하세요. AWSSDKs • AWSAPIs 경우 IAM 사용 설명서의 IAM 사용자의 액세스 키 관리를 참조하세요.

관련 항목:

- IAM 사용 설명서의 IAM이란 무엇입니까?
- AWS일반 참조AWS의 보안 자격 증명.

권한 설정(신규MemoryDB 사용자에게만 해당)

액세스 권한을 제공하려면 사용자, 그룹 또는 역할에 권한을 추가하세요.

- 의 사용자 및 그룹AWS IAM Identity Center:

  권한 세트를 생성합니다. AWS IAM Identity Center 사용자 안내서에서 권한 세트 생성의 지침을 따릅니다.
- ID 제공업체를 통해 IAM에서 관리되는 사용자:

ID 페더레이션을 위한 역할을 생성합니다. IAM 사용자 안내서에서 서드 파티 ID 제공업체의 역할 생성(페더레이션)의 지침을 따릅니다.

- IAM 사용자:
  - 사용자가 맡을 수 있는 역할을 생성합니다. IAM 사용자 안내서에서 IAM 사용자의 역할 생성의 지침을 따릅니다.
  - (권장되지 않음) 정책을 사용자에게 직접 연결하거나 사용자를 사용자 그룹에 추가합니다. IAM 사용 설명서에서 사용자(콘솔)에 권한 추가의 지침을 따릅니다.

MemoryDB에서는 서비스 연결 역할을 생성 및 사용해 리소스를 프로비저닝하고 사용자를 대신해 다른 AWS 리소스 및 서비스에 액세스합니다. MemoryDB에서 서비스 연결 역할을 생성하려면 라는 AWS관리형 정책을 사용합니다`AmazonMemoryDBFullAccess`. 이 역할은 서비스가 사용자를 대신해 서비스 연결 역할을 생성하는 데 필요한 권한으로 사전에 프로비저닝되어 있습니다.

사용자는 기본 정책을 사용하는 대신 사용자 지정 관리형 정책을 사용하는 쪽을 선택할 수 있습니다. 이 경우 `iam:createServiceLinkedRole`을(를) 직접적으로 호출할 권한을 갖고 있거나 MemoryDB 서비스 연결 역할을 생성했어야 합니다.

자세한 내용은 다음을 참조하세요.

- 새 정책 생성(IAM)
- MemoryDB에 대한AWS관리형 (사전 정의된) 정책
- MemoryDB에 대한 서비스 연결 역할 사용

AWSCLI 다운로드 및 구성

AWS CLI는 http://aws.amazon.com/cli 사용할 수 있습니다. Windows, MacOS 및 Linux에서 실행됩니다. 를 다운로드한 후 다음 단계에 AWS CLI따라 설치 및 구성합니다.

1. AWS Command Line Interface 사용 설명서로 이동합니다.
2. AWSCLI 설치 및 AWSCLI 구성에 대한 지침을 따릅니다.

2단계: 클러스터 생성

프로덕션 용도로 사용할 클러스터를 생성하기 전에 비즈니스 요구 사항에 맞게 클러스터를 구성할 방법을 고려해야 합니다. 이러한 문제에 대해서는 클러스터 준비 섹션에서 다룹니다. 이 시작하기 연습의 목적에 따라, 적용되는 모든 위치에서 기본 구성 값을 그대로 사용할 수 있습니다.

생성하려는 클러스터는 활성화되고 샌드박스에서 실행되지 않습니다. 삭제하기 전까지 인스턴스에 대해 표준 MemoryDB 사용 요금이 청구됩니다. 여기에 설명된 연습을 한 번에 끝내고 연습을 마칠 때 클러스터를 삭제하면 총 청구 비용이 가장 적게 듭니다(일반적으로 1달러 미만). MemoryDB 사용 요금에 대한 자세한 내용은 MemoryDB를 참조하세요.

Amazon VPC 서비스 기반의 가상 프라이빗 클라우드(VPC)에서 클러스터를 시작합니다.

MemoryDB 클러스터 생성

다음 예제에서는 AWS Management 콘솔AWS CLI및 MemoryDB API를 사용하여 클러스터를 생성하는 방법을 보여줍니다.

클러스터 생성(콘솔)

MemoryDB 콘솔을 사용하여 클러스터를 생성하려면

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/memorydb/ MemoryDB 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 클러스터를 선택하고 생성을 선택합니다.

Easy create

1. 구성 섹션을 완료합니다. 이렇게 하면 클러스터의 노드 유형과 기본 구성이 구성됩니다. 다음 옵션 중에서 필요한 적절한 메모리 크기와 네트워크 성능을 선택합니다.
    - 프로덕션
    - 개발 및 테스트
    - 데모
2. 클러스터 정보 섹션을 작성하세요.
    a. 이름에 클러스터의 이름을 입력합니다.

       클러스터 명명 제약 조건은 다음과 같습니다.

        - 1~40자의 영숫자 또는 하이픈으로 구성되어야 합니다.
        - 문자로 시작해야 합니다.
        - 하이픈 2개가 연속될 수 없습니다.
        - 끝에 하이픈이 올 수 없습니다.
    b. 설명 상자에 이 클러스터에 대한 설명을 입력합니다.

3. 서브넷 그룹 섹션을 완료하세요.

    - 서브넷 그룹에서 새 서브넷 그룹을 만들거나 사용 가능한 목록에서 이 클러스터에 적용할 기존 서브넷 그룹을 선택합니다. 새로 만드는 경우:

        - 이름을 입력합니다.
        - 설명을 입력합니다.
        - 다중 AZ를 활성화한 경우, 서브넷 그룹에는 서로 다른 가용 영역에 상주하는 서브넷이 두 개 이상 있어야 합니다. 자세한 내용은 서브넷 및 서브넷 그룹 단원을 참조하십시오.
        - 새 서브넷 그룹을 만들고 기존 VPC가 없는 경우, VPC를 생성하라는 메시지가 표시됩니다. 자세한 내용은 Amazon VPC 사용 설명서의 Amazon VPC란 무엇인가요?를 참조하세요.

4. 벡터 검색의 경우 벡터 검색 기능을 활성화하여 벡터 임베딩을 저장하고 벡터 검색을 수행할 수 있습니다. 단, 이렇게 하면 엔진 버전 호환성, 파라미터 그룹 및 샤드의 값이 수정됩니다. 자세한 내용은 벡터 검색 단원을 참조하십시오.

5. 기본 설정 보기:

    간편 생성을 사용하는 경우 나머지 클러스터 설정은 기본적으로 설정됩니다. 이러한 설정 중 일부는 생성 후 편집 가능 표시에 나타난 대로 생성 후에도 변경할 수 있습니다.

6. 태그의 경우 선택적으로 태그를 적용하여 클러스터를 검색 및 필터링하거나 AWS비용을 추적할 수 있습니다.

7. 입력 및 선택한 내용을 모두 검토한 다음 필요한 내용을 수정합니다. 준비가 되면 생성을 선택하여 클러스터를 시작하거나 취소를 선택해 작업을 취소합니다.

클러스터 상태가 사용 가능이 되면 클러스터에 EC2 액세스 권한을 부여하고 클러스터에 연결하며 사용할 수 있습니다. 자세한 내용은 3단계: 클러스터에 대한 액세스 허가 단원을 참조하세요.

> **Important**
>
> 클러스터를 사용할 수 있게 되면 클러스터를 적극 사용하지 않더라도 클러스터가 활성화되어 있는 매 시간 또는 60분 미만 단위로 비용이 청구됩니다. 이 클러스터의 요금 발생을 중지하려면 클러스터를 삭제해야 합니다. 5단계: 클러스터 삭제을(를) 참조하세요.

Create new cluster

1. 클러스터 정보 섹션을 작성하세요.

   a. 이름에 클러스터의 이름을 입력합니다.

      클러스터 명명 제약 조건은 다음과 같습니다.

      - 1~40자의 영숫자 또는 하이픈으로 구성되어야 합니다.
      - 문자로 시작해야 합니다.
      - 하이픈 2개가 연속될 수 없습니다.
      - 끝에 하이픈이 올 수 없습니다.

   b. 설명 상자에 이 클러스터에 대한 설명을 입력합니다.

2. 서브넷 그룹 섹션을 완료하세요.

   - 서브넷 그룹에서 새 서브넷 그룹을 만들거나 사용 가능한 목록에서 이 클러스터에 적용할 기존 서브넷 그룹을 선택합니다. 새로 만드는 경우:

     - 이름을 입력합니다.
     - 설명을 입력합니다.
     - 다중 AZ를 활성화한 경우, 서브넷 그룹에는 서로 다른 가용 영역에 상주하는 서브넷이 두 개 이상 있어야 합니다. 자세한 내용은 서브넷 및 서브넷 그룹 단원을 참조하십시오.
     - 새 서브넷 그룹을 만들고 기존 VPC가 없는 경우, VPC를 생성하라는 메시지가 표시됩니다. 자세한 내용은 Amazon VPC 사용 설명서의 Amazon VPC란 무엇인가요?를 참조하세요.

3. 클러스터 설정 단원을 완료합니다.

   a. 벡터 검색 기능을 활성화하여 벡터 임베딩을 저장하고 벡터 검색을 수행할 수 있습니다. 단, 이렇게 하면 엔진 버전 호환성, 파라미터 그룹 및 샤드의 값이 수정됩니다. 자세한 내용은 벡터 검색 단원을 참조하십시오.

b. 엔진 버전 호환성을 위해 기본값을 그대로 사용하세요. 예를 들어 Valkey의 기본값은 7.2.6이고 Redis OSS의 기본값은 `6.2`입니다.

c. 포트의 경우, 기본 포트인 6379를 그대로 사용하거나, 다른 포트를 사용해야 하는 이유가 있는 경우 포트 번호를 입력합니다.

d. 파라미터 그룹의 경우 벡터 검색을 활성화한 경우 `default.memorydb-valkey7.search`를 사용하세요. 그렇지 않으면 Valkey의 경우 `default.memorydb-valkey7` 파라미터 그룹을 수락하세요.

   파라미터 그룹은 클러스터의 런타임 파라미터를 제어합니다. 파라미터 그룹에 대한 자세한 정보는 [엔진별 파라미터](#) 단원을 참조하세요.

e. 노드 유형에서 원하는 노드 유형 값(관련 메모리 크기 포함)을 선택합니다.

   r6gd 패밀리의 노드 유형을 선택하면 자동으로 데이터 계층화를 활성화하여 메모리와 SSD 간에 데이터 스토리지를 분할합니다. 자세한 내용은 [데이터 계층화](#) 단원을 참조하십시오.

f. 샤드 수에서 이 클러스터에 사용할 샤드 수를 선택합니다. 클러스터의 가용성을 높이려면 샤드를 2개 이상 추가하는 것이 좋습니다.

   클러스터의 샤드 수를 동적으로 변경할 수 있습니다. 자세한 내용은 [MemoryDB 클러스터 크기 조정](#) 단원을 참조하십시오.

g. 샤드당 복제본에서 각 샤드에 포함할 읽기 전용 복제본 노드 수를 선택합니다.

   다음과 같은 제한 사항이 있습니다.

   - 다중 AZ를 활성화한 경우 샤드당 복제본이 하나 이상 있어야 합니다.
   - 콘솔을 사용하여 클러스터를 생성할 때 샤드마다 복제본 수가 동일합니다.

h. 다음을 선택합니다.

i. 고급 설정 섹션을 완료하세요.

   i. 보안 그룹에서 이 클러스터에 사용할 보안 그룹을 선택합니다. 보안 그룹은 클러스터에 대한 네트워크 액세스를 제어하는 방화벽 역할을 합니다. VPC의 기본 보안 그룹을 사용하거나 새 보안 그룹을 만들 수 있습니다.

      보안 그룹에 대한 자세한 정보는 Amazon VPC 사용 설명서의 [VPC의 보안 그룹](#)을 참조하세요.

   ii. 데이터를 암호화하려면 다음과 같은 옵션이 있습니다.

- 저장된 데이터 암호화 - 디스크에 저장된 데이터 암호화를 활성화합니다. 자세한 정보는 [저장된 데이터 암호화]를 참조하세요.

> **Note**
>
> 고객 관리형 AWS소유 KMS 키를 선택하고 키를 선택하여 기본값 이외의 암호화 키를 제공할 수 있습니다.

- 전송 중 데이터 암호화 – 전송 데이터 암호화를 활성화합니다. 암호화를 선택하지 않으면 "오픈 액세스"라는 개방형 액세스 제어 목록이 기본 사용자와 함께 생성됩니다. 자세한 내용은 [액세스 제어 목록(ACL)을 사용하여 사용자 인증] 단원을 참조하십시오.

iii. 스냅샷의 경우, 스냅샷 보존 기간과 스냅샷 기간을 선택적으로 지정합니다. 기본적으로 자동 스냅샷 활성화가 미리 선택되어 있습니다.

iv. 유지 관리 창의 경우, 선택적으로 유지 관리 기간을 지정할 수 있습니다. 유지 관리 기간은 MemoryDB가 클러스터의 시스템 유지 관리를 예약하는 시간이며 일반적으로 매주 한 시간입니다. MemoryDB에서 유지 관리 기간의 요일과 시간을 선택하도록 허용하거나(기본 설정 없음) 요일 시간 및 기간을 직접 선택할 수 있습니다(유지 관리 기간 지정). [Specify maintenance window]를 선택할 경우 목록에서 유지 관리 기간의 [Start day], [Start time] 및 [Duration](시간)을 선택합니다. 모든 시간은 UCT 시간입니다.

    자세한 내용은 [유지 관리 관리 중] 단원을 참조하십시오.

v. 알림에 대해 기존의 Amazon Simple Notification Service(Amazon SNS) 항목을 선택하거나 수동 ARN 입력을 선택하고 Amazon 리소스 이름(ARN) 항목을 입력합니다. Amazon SNS를 통해 인터넷에 연결된 스마트 디바이스에 알림을 푸시할 수 있습니다. 기본적으로 알림이 비활성화됩니다. 자세한 내용은 https://aws.amazon.com/sns/를 참조하세요.

vi. 태그의 경우 선택적으로 태그를 적용하여 클러스터를 검색 및 필터링하거나 AWS비용을 추적할 수 있습니다.

j. 입력 및 선택한 내용을 모두 검토한 다음 필요한 내용을 수정합니다. 준비가 되면 생성을 선택하여 클러스터를 시작하거나 취소를 선택해 작업을 취소합니다.

클러스터 상태가 사용 가능이 되면 클러스터에 EC2 액세스 권한을 부여하고 클러스터에 연결하며 사용할 수 있습니다. 자세한 내용은 3단계: 클러스터에 대한 액세스 허가 단원을 참조하세요.

> **Important**
>
> 클러스터를 사용할 수 있게 되면 클러스터를 적극 사용하지 않더라도 클러스터가 활성화되어 있는 매 시간 또는 60분 미만 단위로 비용이 청구됩니다. 이 클러스터의 요금 발생을 중지하려면 클러스터를 삭제해야 합니다. 5단계: 클러스터 삭제을(를) 참조하세요.

Restore from snapshots

스냅샷 소스에서 데이터를 마이그레이션할 소스 스냅샷을 선택합니다. 자세한 내용은 스냅샷 및 복원 단원을 참조하십시오.

> **Note**
>
> 새 클러스터에 벡터 검색을 활성화하려면 소스 스냅샷에도 벡터 검색이 활성화되어 있어야 합니다.

타겟 클러스터의 기본값은 소스 클러스터의 설정입니다. 선택적으로 대상 클러스터에서 다음 설정을 변경할 수 있습니다.

1. 클러스터 정보

   a. 이름에 클러스터의 이름을 입력합니다.

      클러스터 명명 제약 조건은 다음과 같습니다.

      - 1~40자의 영숫자 또는 하이픈으로 구성되어야 합니다.
      - 문자로 시작해야 합니다.
      - 하이픈 2개가 연속될 수 없습니다.
      - 끝에 하이픈이 올 수 없습니다.

   b. 설명 상자에 이 클러스터에 대한 설명을 입력합니다.

2. 서브넷 그룹

   - 서브넷 그룹에서 새 서브넷 그룹을 만들거나 사용 가능한 목록에서 이 클러스터에 적용할 기존 서브넷 그룹을 선택합니다. 새로 만드는 경우:

     - 이름을 입력합니다.
     - 설명을 입력합니다.
     - 다중 AZ를 활성화한 경우, 서브넷 그룹에는 서로 다른 가용 영역에 상주하는 서브넷이 두 개 이상 있어야 합니다. 자세한 내용은 서브넷 및 서브넷 그룹 단원을 참조하십시오.
     - 새 서브넷 그룹을 만들고 기존 VPC가 없는 경우, VPC를 생성하라는 메시지가 표시됩니다. 자세한 내용은 Amazon VPC 사용 설명서의 Amazon VPC란 무엇인가요?를 참조하세요.

3. 클러스터 설정

   a. 벡터 검색 기능을 활성화하여 벡터 임베딩을 저장하고 벡터 검색을 수행할 수 있습니다. 단, 이렇게 하면 엔진 버전 호환성, 파라미터 그룹 및 샤드의 값이 수정됩니다. 자세한 내용은 벡터 검색 단원을 참조하십시오.

   b. 엔진 버전 호환성을 위해 기본값 `6.2`를 그대로 사용하세요.

   c. 포트의 경우, 기본 포트인 6379를 그대로 사용하거나, 다른 포트를 사용해야 하는 이유가 있는 경우 포트 번호를 입력합니다.

   d. 파라미터 그룹의 경우 벡터 검색을 활성화한 경우 `default.memorydb-redis7.search.preview`를 사용하세요. 그렇지 않으면 `default.memorydb-redis7` 파라미터 그룹을 수락하세요.

      파라미터 그룹은 클러스터의 런타임 파라미터를 제어합니다. 파라미터 그룹에 대한 자세한 정보는 엔진별 파라미터 단원을 참조하세요.

   e. 노드 유형에서 원하는 노드 유형 값(관련 메모리 크기 포함)을 선택합니다.

      r6gd 패밀리의 노드 유형을 선택하면 자동으로 데이터 계층화를 활성화하여 메모리와 SSD 간에 데이터 스토리지를 분할합니다. 자세한 내용은 데이터 계층화 단원을 참조하십시오.

   f. 샤드 수에서 이 클러스터에 사용할 샤드 수를 선택합니다. 클러스터의 가용성을 높이려면 샤드를 2개 이상 추가하는 것이 좋습니다.

      클러스터의 샤드 수를 동적으로 변경할 수 있습니다. 자세한 내용은 MemoryDB 클러스터 크기 조정 단원을 참조하십시오.

g. 샤드당 복제본에서 각 샤드에 포함할 읽기 전용 복제본 노드 수를 선택합니다.

   다음과 같은 제한 사항이 있습니다.

   - 다중 AZ를 활성화한 경우 샤드당 복제본이 하나 이상 있어야 합니다.
   - 콘솔을 사용하여 클러스터를 생성할 때 샤드마다 복제본 수가 동일합니다.

h. 다음을 선택합니다.

i. 고급 설정

   i. 보안 그룹에서 이 클러스터에 사용할 보안 그룹을 선택합니다. 보안 그룹은 클러스터에 대한 네트워크 액세스를 제어하는 방화벽 역할을 합니다. VPC의 기본 보안 그룹을 사용하거나 새 보안 그룹을 만들 수 있습니다.

      보안 그룹에 대한 자세한 정보는 Amazon VPC 사용 설명서의 VPC의 보안 그룹을 참조하세요.

   ii. 데이터를 암호화하려면 다음과 같은 옵션이 있습니다.

      - 저장된 데이터 암호화 - 디스크에 저장된 데이터 암호화를 활성화합니다. 자세한 정보는 저장된 데이터 암호화를 참조하세요.

        > **Note**
        >
        > 고객 관리형 AWS소유 KMS 키를 선택하고 키를 선택하여 기본값 이외의 암호화 키를 제공할 수 있습니다.

      - 전송 중 데이터 암호화 – 전송 데이터 암호화를 활성화합니다. 암호화를 선택하지 않으면 “오픈 액세스”라는 개방형 액세스 제어 목록이 기본 사용자와 함께 생성됩니다. 자세한 내용은 액세스 제어 목록(ACL)을 사용하여 사용자 인증 단원을 참조하십시오.

   iii. 스냅샷의 경우, 스냅샷 보존 기간과 스냅샷 기간을 선택적으로 지정합니다. 기본적으로 자동 스냅샷 활성화가 미리 선택되어 있습니다.

   iv. 유지 관리 창의 경우, 선택적으로 유지 관리 기간을 지정할 수 있습니다. 유지 관리 기간은 MemoryDB가 클러스터의 시스템 유지 관리를 예약하는 시간이며 일반적으로 매주 한 시간입니다. MemoryDB에서 유지 관리 기간의 요일과 시간을 선택하도록 허용하거나(기본 설정 없음) 요일 시간 및 기간을 직접 선택할 수 있습니다(유지 관리 기간 지정). [Specify maintenance window]를 선택할 경우 목록에서 유지 관리 기간의

[Start day], [Start time] 및 [Duration](시간)을 선택합니다. 모든 시간은 UCT 시간입니다.

자세한 내용은 유지 관리 관리 중 단원을 참조하십시오.

v. 알림에 대해 기존의 Amazon Simple Notification Service(Amazon SNS) 항목을 선택하거나 수동 ARN 입력을 선택하고 Amazon 리소스 이름(ARN) 항목을 입력합니다. Amazon SNS를 통해 인터넷에 연결된 스마트 디바이스에 알림을 푸시할 수 있습니다. 기본적으로 알림이 비활성화됩니다. 자세한 내용은 https://aws.amazon.com/sns/를 참조하세요.

vi. 태그의 경우 선택적으로 태그를 적용하여 클러스터를 검색 및 필터링하거나 AWS비용을 추적할 수 있습니다.

j. 입력 및 선택한 내용을 모두 검토한 다음 필요한 내용을 수정합니다. 준비가 되면 생성을 선택하여 클러스터를 시작하거나 취소를 선택해 작업을 취소합니다.

클러스터 상태가 사용 가능이 되면 클러스터에 EC2 액세스 권한을 부여하고 클러스터에 연결하며 사용할 수 있습니다. 자세한 내용은 3단계: 클러스터에 대한 액세스 허가 단원을 참조하세요.

> **Important**
>
> 클러스터를 사용할 수 있게 되면 클러스터를 적극 사용하지 않더라도 클러스터가 활성화되어 있는 매 시간 또는 60분 미만 단위로 비용이 청구됩니다. 이 클러스터의 요금 발생을 중지하려면 클러스터를 삭제해야 합니다. 5단계: 클러스터 삭제을(를) 참조하세요.

클러스터 생성(AWSCLI)

를 사용하여 클러스터를 생성하려면 섹션을 AWS CLI참조하세요create-cluster. 다음은 예제입니다.

Linux, macOS 또는 Unix의 경우는 다음과 같습니다.

```
aws memorydb create-cluster \
    --cluster-name my-cluster \
    --node-type db.r6g.large \
    --acl-name my-acl \
    --engine valkey \
    --subnet-group my-sg
```

Windows의 경우:

```
aws memorydb create-cluster ^
   --cluster-name my-cluster ^
   --node-type db.r6g.large ^
   --acl-name my-acl ^
   --engine valkey
   --subnet-group my-sg
```

다음과 같은 JSON 응답을 받아야 합니다.

```
{
    "Cluster": {
        "Name": "my-cluster",
        "Status": "creating",
        "NumberOfShards": 1,
        "AvailabilityMode": "MultiAZ",
        "ClusterEndpoint": {
            "Port": 6379
        },
        "NodeType": "db.r6g.large",
        "EngineVersion": "7.2",
        "EnginePatchVersion": "7.2.6",
        "ParameterGroupName": "default.memorydb-valkey7",
        "Engine": "valkey"
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
```

```
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxxxxxxxxx:cluster/my-cluster",
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
        "SnapshotWindow": "04:30-05:30",
        "ACLName": "my-acl",
        "DataTiering": "false",
        "AutoMinorVersionUpgrade": true
    }
}
```

상태가 `available`로 변경되면 클러스터 사용을 시작할 수 있습니다.

> **Important**
>
> 클러스터를 사용할 수 있게 되면 클러스터를 적극 사용하지 않더라도 클러스터가 활성화되어 있는 매 시간 또는 60분 미만 단위로 비용이 청구됩니다. 이 클러스터의 요금 발생을 중지하려면 클러스터를 삭제해야 합니다. 5단계: 클러스터 삭제을(를) 참조하세요.

클러스터 생성(MemoryDB API)

MemoryDB API를 사용하여 클러스터를 생성하려면 CreateCluster 작업을 사용하세요.

> **Important**
>
> 클러스터를 사용할 수 있게 되면 클러스터를 사용하지 않더라도 클러스터가 활성화되어 있는 매 시간 또는 60분 미만 단위로 비용이 청구됩니다. 이 클러스터의 요금 발생을 중지하려면 클러스터를 삭제해야 합니다. 5단계: 클러스터 삭제을(를) 참조하세요.

인증 설정

클러스터의 인증 설정에 대한 자세한 내용은 IAM을 통한 인증 및 액세스 제어 목록(ACL)을 사용하여 사용자 인증을 참조하세요.

3단계: 클러스터에 대한 액세스 허가

이 섹션에서는 Amazon EC2 인스턴스를 시작하고 연결하는 것에 익숙하다고 가정합니다. 자세한 내용은 Amazon EC2 시작 안내서를 참조하세요.

모든 MemoryDB 클러스터는 Amazon EC2 인스턴스에서 액세스하도록 설계되었습니다. Amazon Elastic Container Service 또는 AWS Lambda에서 실행되는 컨테이너식 또는 서버리스 애플리케이션에서도 액세스할 수 있습니다. 가장 일반적인 시나리오는 동일한 Amazon Virtual Private Cloud(Amazon VPC)의 Amazon EC2 인스턴스에서 MemoryDB 클러스터에 액세스하는 것이며, 이 연습의 사례도 이에 해당합니다.

EC2 인스턴스에서 클러스터에 연결하려면 EC2 인스턴스가 클러스터에 액세스하도록 권한을 부여해야 합니다.

가장 일반적인 사용 사례는 EC2 인스턴스에 배포된 애플리케이션이 같은 VPC에 있는 클러스터에 연결해야 하는 경우입니다. 동일한 VPC에서 EC2 인스턴스와 클러스터 간 액세스를 관리하는 가장 간단한 방법은 다음과 같습니다.

1. 클러스터의 VPC 보안 그룹을 만듭니다. 이 보안 그룹을 사용해 클러스터에 대한 액세스를 제한할 수 있습니다. 예를 들어, 클러스터를 만들 때 할당한 포트와 클러스터에 액세스할 때 이용할 IP 주소를 사용해 TCP 액세스를 허용하는 이 보안 그룹의 사용자 지정 규칙을 만들 수 있습니다.

   MemoryDB 클러스터의 기본 포트는 6379입니다.
2. EC2 인스턴스(웹 및 애플리케이션 서버)의 VPC 보안 그룹을 만듭니다. 이 보안 그룹은 필요할 경우 VPC의 라우팅 테이블을 통한 EC2 인스턴스 액세스를 허용할 수 있습니다. 예를 들어, 이 보안 그룹에서 TCP가 포트 22를 통해 EC2 인스턴스에 액세스하도록 허용하는 규칙을 설정할 수 있습니다.
3. 클러스터에 대한 보안 그룹에서 EC2 인스턴스에 대해 생성한 보안 그룹으로부터의 연결을 허용하는 사용자 지정 규칙을 만듭니다. 그러면 보안 그룹의 모든 구성원이 클러스터에 액세스하도록 허용됩니다.

VPC 보안 그룹에서 다른 보안 그룹으로부터의 연결을 허용하는 규칙을 만들려면

1. AWSManagement Console에 로그인하고 https://console.aws.amazon.com/vpc Amazon VPC 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 [Security Groups]를 선택합니다.

3. 클러스터에 사용할 보안 그룹을 선택하거나 만듭니다. [Inbound Rules]에서 [Edit Inbound Rules]를 선택한 다음 [Add Rule]을 선택합니다. 이 보안 그룹은 다른 보안 그룹 멤버에 대한 액세스를 허용합니다.
4. [Type]에서 [Custom TCP Rule]을 선택합니다.
   a. [Port Range]에 대해 클러스터를 만들 때 사용한 포트를 지정합니다.

      MemoryDB 클러스터의 기본 포트는 6379입니다.
   b. [Source] 상자에 보안 그룹 ID를 입력합니다. 목록에서 Amazon EC2 인스턴스에 사용할 보안 그룹을 선택합니다.
5. 완료되면 [Save]를 선택합니다.

액세스를 활성화했으면 이제 다음 섹션에 설명된 대로 클러스터에 연결할 수 있습니다.

다른 Amazon VPC, 다른 AWS리전 또는 회사 네트워크에서 MemoryDB 클러스터에 액세스하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- Amazon VPC에 있는 MemoryDB 클러스터에 액세스하기 위한 액세스 패턴
- AWS 외부에서 MemoryDB 리소스에 액세스

4단계: 클러스터에 연결

계속하기 전에 3단계: 클러스터에 대한 액세스 허가 단계를 완료하세요.

이 섹션에서는 Amazon EC2 인스턴스를 생성했고 이 인스턴스에 연결할 수 있다고 가정합니다. 작업 방법에 대한 지침은 Amazon EC2 시작 안내서를 참조하세요.

Amazon EC2 인스턴스는 권한을 부여한 경우에만 클러스터에 연결할 수 있습니다.

클러스터 엔드포인트 찾기

클러스터가 사용 가능한 상태이고 사용자가 이 클러스터에 액세스할 수 있는 권한을 부여받았다면 Amazon EC2 인스턴스에 로그인하여 클러스터에 연결할 수 있습니다. 이를 수행하려면 먼저 엔드포인트를 결정해야 합니다.

엔드포인트를 찾는 방법을 더 자세히 알아보려면 다음을 참조하세요.

- MemoryDB 클러스터의 엔드포인트 찾기(AWS Management 콘솔)
- MemoryDB 클러스터(CLI AWS)의 엔드포인트 찾기
- MemoryDB 클러스터의 엔드포인트 찾기(MemoryDB API)

MemoryDB 클러스터에 연결(Linux)

이제 필요한 엔드포인트가 있으므로 EC2 인스턴스에 로그인하여 클러스터에 연결할 수 있습니다. 다음 예제에서는 cli 유틸리티를 사용하여 Ubuntu 22를 사용하는 클러스터에 연결합니다. 최신 버전의 cli는 암호화/인증이 활성화된 클러스터에 연결할 수 있도록 SSL/TLS도 지원합니다.

redis-cli를 사용하여 MemoryDB 노드에 연결

MemoryDB 노드에서 데이터에 액세스하려면 Secure Sockets Layer(SSL) 작업을 하는 클라이언트를 사용해야 합니다. 또한 Amazon Linux 및 Amazon Linux 2에서 TLS/SSL과 함께 redis-cli를 사용할 수 있습니다.

redis-cli를 사용하여 Amazon Linux 2 또는 Amazon Linux에서 MemoryDB 클러스터에 연결하려면

1. redis-cli 유틸리티를 다운로드하고 컴파일합니다. 이 유틸리티는 Redis OSS 소프트웨어 배포에 포함되어 있습니다.
2. EC2 인스턴스의 명령 프롬프트에 사용 중인 Linux 버전에 적합한 명령을 입력합니다.

Amazon Linux 2023

Amazon Linux 2023을 사용하는 경우 다음을 입력합니다.

```
sudo yum install redis6 -y
```

그런 다음, 다음 명령을 입력하고 클러스터 및 포트의 엔드포인트를 이 예제에 표시된 것으로 대체합니다.

```
redis-cli -h Primary or Configuration Endpoint --tls -p 6379
```

엔드포인트 찾기에 대한 자세한 내용은 노드 엔드포인트 찾기 섹션을 참조하세요.

Amazon Linux 2

Amazon Linux 2를 사용하는 경우 다음을 입력합니다.

```
sudo yum -y install openssl-devel gcc
wget https://download.redis.io/releases/redis-7.2.5.tar.gz
tar xvzf redis-7.2.5.tar.gz
cd redis-7.2.5
make distclean
make redis-cli BUILD_TLS=yes
sudo install -m 755 src/redis-cli /usr/local/bin/
```

Amazon Linux

Amazon Linux를 사용하는 경우 다음을 입력합니다.

```
sudo yum install gcc jemalloc-devel openssl-devel tcl tcl-devel clang wget
wget https://download.redis.io/releases/redis-7.2.5.tar.gz
tar xvzf redis-7.2.5.tar.gz
cd redis-7.2.5
make redis-cli CC=clang BUILD_TLS=yes
sudo install -m 755 src/redis-cli /usr/local/bin/
```

Amazon Linux에서 다음 추가 단계를 실행해야 할 수도 있습니다.

```
sudo yum install clang
```

```
CC=clang make
sudo make install
```

3. redis-cli 유틸리티를 다운로드하고 설치한 후에는 선택적 `make-test` 명령을 실행하는 것이 좋습니다.
4. 암호화 및 인증이 활성화된 클러스터에 연결하려면 다음 명령을 입력합니다.

```
redis-cli -h Primary or Configuration Endpoint --tls -a 'your-password' -p 6379
```

> **Note**
>
> Amazon Linux 2023에 redis6을 설치하는 경우 이제 `redis-cli` 대신 `redis6-cli` 명령을 사용할 수 있습니다.
>
> ```
> redis6-cli -h Primary or Configuration Endpoint --tls -p 6379
> ```

5단계: 클러스터 삭제

클러스터가 available 상태면 클러스터를 적극 사용하고 있는지 여부에 관계없이 요금이 부과됩니다. 요금 발생을 중지하려면 클러스터를 삭제하세요.

> **Warning**
>
> - MemoryDB 클러스터를 삭제하는 경우, 수동 스냅샷은 보존됩니다. 클러스터가 삭제되기 전에 최종 스냅샷을 생성할 수도 있습니다. 자동 스냅샷은 보존되지 않습니다. 자세한 내용은 [스냅샷 및 복원](#) 단원을 참조하십시오.
> - 최종 스냅샷을 생성하려면 `CreateSnapshot` 권한이 필요합니다. 이 권한이 없으면 API 호출이 실패하고 `Access Denied` 예외가 발생합니다.

사용AWS Management 콘솔

다음은 배포에서 클러스터 하나를 삭제하는 절차입니다. 클러스터를 여러 개 삭제하려면 삭제할 클러스터마다 절차를 반복하세요. 클러스터 하나를 다 삭제한 후 다른 클러스터 삭제 절차가 시작될 때까지 기다릴 필요는 없습니다.

클러스터 삭제

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/memorydb/ MemoryDB 콘솔을 엽니다.
2. 삭제할 클러스터를 선택하려면 클러스터 목록에서 해당 클러스터 이름 옆의 라디오 버튼을 선택합니다. 이 경우 2단계: 클러스터 생성에서 생성된 클러스터의 이름입니다.
3. 작업에 대해 삭제를 선택합니다.
4. 삭제하기 전에 먼저 클러스터의 스냅샷을 생성할지 여부를 선택한 다음 확인 상자에 `delete`을 입력합니다. 클러스터를 삭제하려면 삭제를 입력하고 클러스터를 유지하려면 취소를 선택합니다.

   [Delete]를 선택한 경우 클러스터 상태가 [deleting]으로 바뀝니다.

클러스터가 클러스터 목록에서 제거되는 즉시 요금 부과가 중단됩니다.

사용AWS CLI

다음 코드는 클러스터 `my-cluster`를 삭제합니다. 이 경우 `my-cluster`를 2단계: 클러스터 생성에서 생성된 클러스터의 이름으로 바꿉니다.

```
aws memorydb delete-cluster --cluster-name my-cluster
```

`delete-cluster` CLI 작업은 클러스터를 하나만 삭제합니다. 클러스터를 여러 개 삭제하려면 삭제할 클러스터마다 `delete-cluster`를 호출하세요. 클러스터 하나를 다 삭제한 후 다른 클러스터를 삭제할 때까지 기다릴 필요는 없습니다.

Linux, macOS, Unix의 경우:

```
aws memorydb delete-cluster \
    --cluster-name my-cluster \
    --region us-east-1
```

Windows의 경우:

```
aws memorydb delete-cluster ^
    --cluster-name my-cluster ^
    --region us-east-1
```

자세한 내용은 `delete-cluster` 단원을 참조하십시오.

MemoryDB API 사용

다음 코드는 클러스터 `my-cluster`를 삭제합니다. 이 경우 `my-cluster`를 2단계: 클러스터 생성에서 생성된 클러스터의 이름으로 바꿉니다.

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DeleteCluster
    &ClusterName=my-cluster
    &Region=us-east-1
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210802T220302Z
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210802T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210802T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

`DeleteCluster` API 작업은 클러스터를 하나만 삭제합니다. 클러스터를 여러 개 삭제하려면 삭제할 클러스터마다 `DeleteCluster`를 호출하세요. 클러스터 하나를 다 삭제한 후 다른 클러스터를 삭제할 때까지 기다릴 필요는 없습니다.

자세한 내용은 DeleteCluster 단원을 참조하세요.

다음 단계

이제 시작하기 연습을 끝마쳤으므로 다음 단원을 참조하여 MemoryDB 및 사용 가능한 도구에 대한 자세한 내용을 살펴볼 수 있습니다.

- 시작하기AWS
- Amazon Web Services용 도구
- AWSCommand Line Interface
- MemoryDB API 참조

노드 관리

노드는 MemoryDB 배포의 가장 작은 구성 요소입니다. 노드는 클러스터에 속하는 샤드에 속합니다. 각 노드는 클러스터가 생성되거나 마지막으로 수정되었을 때 선택한 엔진을 실행합니다. 각 노드에는 고유한 DNS(Domain Name Service) 이름 및 포트가 있습니다. 여러 유형의 MemoryDB 노드가 지원되며, 연결된 메모리 양과 컴퓨팅 파워는 각각 다릅니다.

주제

노드와 관련된 중요한 작업은 다음과 같습니다.

- 클러스터에서 노드 추가/제거
- 스케일링
- 연결 엔드포인트 찾기

MemoryDB 노드 및 샤드

샤드는 노드의 계층적 정렬입니다(각각 클러스터에 래핑). 샤드는 복제를 지원합니다. 샤드에서 노드 하나가 읽기/쓰기 프라이머리 노드로 사용됩니다. 샤드에 있는 다른 모든 노드는 프라이머리 노드의 읽기 전용 복제본 역할을 합니다. MemoryDB는 클러스터 내에서 여러 샤드를 지원합니다. 이 지원으로 인해 MemoryDB 클러스터에서 데이터를 파티셔닝할 수 있습니다.

MemoryDB는 샤드를 통한 복제를 지원합니다. API 작업 DescribeClusters는 샤드를 멤버 노드, 노드 이름, 엔드포인트 및 기타 정보와 함께 나열합니다.

MemoryDB 클러스터가 생성된 후에는 변경(스케일 인 또는 스케일 아웃)이 가능합니다. 자세한 내용은 스케일링 및 노드 교체(을)를 참조하세요.

새 클러스터를 생성할 때 빈 상태로 시작되지 않도록 이전 클러스터의 데이터를 시드할 수 있습니다. 이렇게 하면 노드 유형이나 엔진 버전을 변경해야 하는 경우나 Amazon ElastiCache(Redis OSS)에서 마이그레이션해야 하는 경우에 도움이 됩니다. 자세한 내용은 수동 스냅샷 생성 및 스냅샷에서 복원(을)를 참조하세요.

지원되는 노드 유형

MemoryDB는 다음 노드 유형을 지원합니다.

메모리 최적화

인스턴스 유형	기준 대역폭(Gbps)	버스트 대역폭(Gbps)	향상된 I/O 멀티플렉싱(Valkey 7.2 및 Redis OSS 7.0.4 이상)	최소 엔진 버전
db.r7g.large	0.937	12.5	아니요	6.2
db.r7g.xlarge	1.876	12.5	아니요	6.2
db.r7g.2xlarge	3.75	15	예	6.2
db.r7g.4xlarge	7.5	15	예	6.2
db.r7g.8xlarge	15	N/A	예	6.2
db.r7g.12xlarge	22.5	N/A	예	6.2
db.r7g.16xlarge	30	N/A	예	6.2
db.r6g.large	0.75	10.0	아니요	6.2
db.r6g.xlarge	1.25	10.0	아니요	6.2
db.r6g.2xlarge	2.5	10.0	예	6.2
db.r6g.4xlarge	5.0	10.0	예	6.2
db.r6g.8xlarge	12	N/A	예	6.2
db.r6g.12xlarge	20	N/A	예	6.2
db.r6g.16xlarge	25	N/A	예	6.2

데이터 계층화에 최적화된 메모리

인스턴스 유형	기준 대역폭(Gbps)	버스트 대역폭(Gbps)	향상된 I/O 멀티플렉싱(Valkey 7.2 및 Redis OSS 7.0.4 이상)	최소 엔진 버전
db.r6gd.xlarge	1.25	10	아니요	6.2
db.r6gd.2xlarge	2.5	10	아니요	6.2
db.r6gd.4xlarge	5.0	10	아니요	6.2
db.r6gd.8xlarge	12	N/A	아니요	6.2

범용 노드

인스턴스 유형	기준 대역폭(Gbps)	버스트 대역폭(Gbps)	향상된 I/O 멀티플렉싱(Valkey 7.2 및 Redis OSS 7.0.4 이상)	최소 엔진 버전
db.t4g.small	0.128	5.0	아니요	6.2
db.t4g.medium	0.256	5.0	아니요	6.2

AWS 리전 가용성에 대한 내용은 MemoryDB 요금을 참조하세요.

모든 노드 유형은 가상 프라이빗 클라우드(VPC)에 생성됩니다.

MemoryDB 예약 노드

예약 노드는 온디맨드 노드 요금과 비교하여 대폭 할인된 요금을 제공합니다. 예약 노드는 물리적 노드가 아니며 계정에서 온디맨드 노드를 사용할 때 적용되는 결제 할인에 가깝습니다. 예약 노드 할인은 노드 유형 및 AWS 리전에 따라 다릅니다.

> **Note**
>
> 모든 현재 MemoryDB 예약 노드는 Redis OSS 엔진을 실행하는 노드의 요금을 기준으로 하며 해당 노드의 담당률을 제공합니다. 이러한 예약 노드는 유연한 크기의 예약 노드에 설명된 대로 Valkey 엔진에 적용할 수 있지만 Valkey 전용 예약 노드는 사용할 수 없습니다.

예약 노드를 사용하는 일반적인 프로세스는 다음과 같습니다.

- 사용 가능한 예약 노드 제품에 대한 정보를 검토하려면
- AWS Management 콘솔 및 AWS Command Line Interface 또는 SDK를 사용하여 예약 노드를 구매합니다.
- 기존 예약 노드에 대한 정보를 검토하세요.

주제

예약 노드 개요

MemoryDB 예약 노드를 구매할 때는 예약 노드의 기간 동안 특정 노드 유형에 대해 할인 요금을 이용하는 약정을 구매하는 것입니다. MemoryDB 예약 노드를 사용하려면 온디맨드 노드를 생성하는 것과 똑같은 방법으로 새로운 노드를 생성해야 합니다. 새 노드는 예약 노드의 사양과 일치해야 합니다. 새로운 노드의 사양이 계정의 기존 예약 노드와 일치하면 예약 노드에 제공되는 할인 요금이 청구됩니

다. 그렇지 않으면 노드에 대해 온디맨드 요금이 청구됩니다. AWS Management 콘솔, AWS CLI 또는 MemoryDB API를 사용하여 사용 가능한 예약 노드 제품을 나열하고 구입할 수 있습니다.

MemoryDB는 메모리에 최적화된 R7g, R6g 및 R6gd(데이터 계층화 포함) 노드용 예약 노드를 제공합니다. 요금에 대한 자세한 내용은 [MemoryDB 요금]을 참조하세요.

제공 유형

예약 노드는 세 가지 유형(No Upfront, Partial Upfront 및 All Upfront)으로 제공되며 예상되는 사용률에 따라 MemoryDB 비용을 최적화할 수 있습니다.

비선결제 – 선결제 없이 예약 노드에 액세스할 수 있는 옵션입니다. 비선결제 예약 노드는 사용 기간 동안 사용량에 상관없이 할인된 시간당 요금이 청구되며, 선결제가 필요하지 않습니다.

부분 선결제 - 예약 노드 사용비의 일부를 먼저 결제해야 하는 옵션입니다. 결제하지 않은 시간에 대해서는 사용 기간 동안 사용량에 상관없이 할인된 시간당 요금이 청구됩니다.

전액 선불 – 약관이 시작되는 시점에서 모든 금액을 결제하고 사용 기간 동안 추가 비용 없이 무제한으로 사용할 수 있습니다.

세 가지 제공 유형 모두 1년과 3년 단위로 사용할 수 있습니다.

유연한 크기의 예약 노드

예약 노드를 구매할 때 지정하는 한 가지 사항은 노드 유형(예: db.r6g.xlarge)입니다. 노드 유형에 대한 자세한 내용은 [MemoryDB 요금]을 참조하세요.

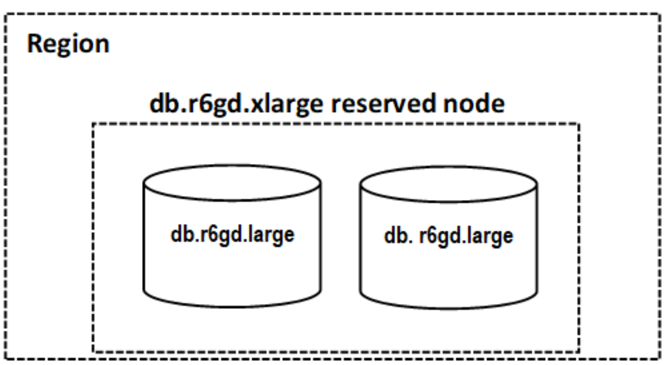
이미 노드가 있지만 용량을 확장해야 하는 경우에는 예약 노드가 확장된 노드에 자동으로 적용됩니다. 즉, 예약 노드는 동일한 노드 패밀리의 모든 규모의 사용에 자동으로 적용됩니다. 크기가 유연한 예약 노드는 동일한 지역의 AWS 노드에 사용할 수 있습니다. 유연한 크기의 예약 노드는 해당 노드 제품군에서만 확장할 수 있습니다. 예를 들어 db.r6g.xlarge의 예약 노드는 b.r6g.2xlarge에 적용할 수 있지만, db.r6gd.large에는 적용할 수 없습니다. db.r6g과 db.r6gd는 다른 노드 클래스 유형이기 때문입니다.

크기 유연성은 동일한 노드 클래스 유형 내에서 구성 간에 자유롭게 이동할 수 있음을 의미합니다. 예를 들어, 추가 비용 없이 동일한 AWS 지역의 r6g.xlarge 예약 노드 1개(정규화된 유닛 8개)에서 r6g.large 예약 노드 2개(정규화된 유닛 8개) (2*4 = 정규화된 유닛 8개)로 이동할 수 있습니다.

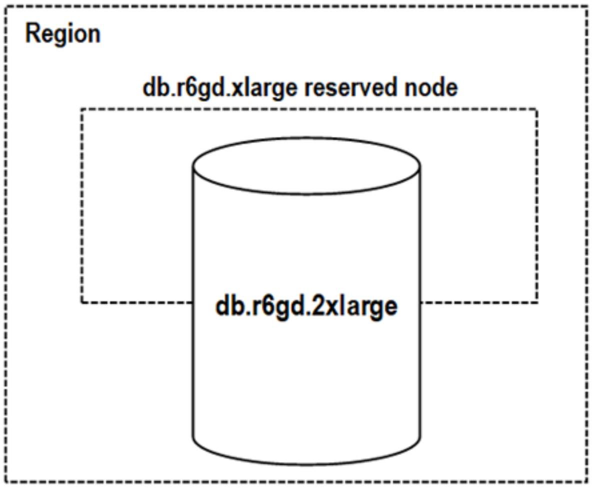
예약 노드의 크기에 따른 사용량은 정규화 유닛을 사용하여 비교할 수 있습니다. 예를 들어, 두 개의 db.r6g.4xlarge 노드에서 1시간을 사용하면 하나의 db.r6g.large에서 16시간을 사용하는 것과 같습니다. 다음 표는 노드 크기에 따른 정규화 유닛의 수를 나타낸 것입니다.

노드 크기	정규화된 단위(Redis OSS)	정규화된 단위(Valkey)
small	1	.7
medium	2	1.4
large	4	2.8
xlarge	8	5.6
2xlarge	16	11.2
4xlarge	32	22.4
6xlarge	48	33.6
8xlarge	64	44.8
10xlarge	80	56
12xlarge	96	67.2
16xlarge	128	89.6
24xlarge	192	134.4

db.r6gd.xlarge 예약 노드를 1개 구매하고, 동일한 AWS 리전의 계정에서 db.r6gd.large 예약 노드를 2개 실행하는 경우를 예로 들어 보겠습니다. 이 경우, 결제 혜택은 두 노드에 100% 적용됩니다.



또는 동일한 AWS 리전의 계정에서 실행 중인 db.r6gd.2xlarge 인스턴스 1개가 있는 경우, 결제 혜택은 노드 사용량의 50%에 적용됩니다.



노드를 Redis OSS에서 Valkey로 업그레이드

MemoryDB에서 Valkey를 시작하면 이제 Redis OSS 예약 노드 할인을 Valkey 엔진에 적용할 수 있습니다. 기존 계약 및 예약의 이점을 유지하면서 Redis OSS에서 Valkey로 업그레이드할 수 있습니다. 노드 패밀리 및 엔진 내에서 이점을 적용할 수 있을 뿐만 아니라 더 많은 증분 값을 얻을 수도 있습니다. Valkey는 Redis OSS에 비해 30% 할인된 가격으로 제공되며, 예약된 노드 유연성을 통해 Redis OSS 예약 노드를 사용하여 실행 중인 Valkey 노드를 더 많이 커버할 수 있습니다.

할인율을 계산하기 위해 각 MemoryDB 노드와 엔진 조합에는 단위로 측정되는 정규화 인수가 있습니다. 예약 노드 단위는 지정된 엔진에 대해 예약 노드의 인스턴스 패밀리 내에 있는 실행 중인 모든 노드에 적용할 수 있습니다. Redis OSS 예약 노드는 엔진에 추가로 적용하여 실행 중인 Valkey 노드를 커버할 수 있습니다. Valkey는 Redis OSS에 비해 할인된 가격으로 가격이 책정되기 때문에 지정된 인스턴스 유형에 대한 단위가 낮아 Redis OSS 예약 노드가 더 많은 Valkey 노드를 커버할 수 있습니다.

예를 들어, Redis OSS 엔진(32 단위)에 대해 db.r7g.4xlarge용 예약 노드를 구매했고 db.r7g.4xlarge Redis OSS 노드(32 단위) 하나를 실행 중이라고 가정해 보겠습니다. 노드를 Valkey로 업그레이드하면 실행 중인 노드의 정규화 인수가 22.4단위로 떨어지며, 기존 예약 노드는 리전의 db.r7g 패밀리 내에서 실행 중인 다른 Valkey 또는 Redis OSS 노드에 사용할 추가 9.6단위를 제공합니다. 이를 사용하여 계정의 다른 db.r7g.4xlarge Valkey 노드의 42%(22.4 단위) 또는 db.r7g.xlarge Valkey 노드의 100%(5.6 단위) 및 db.r7g.large Valkey 노드의 100%(2.8단위)를 포함할 수 있습니다.

예약 노드 삭제

예약 노드에 대한 약정 기간은 1년 또는 3년입니다. 예약 노드를 취소할 수 없습니다. 하지만 예약 노드 할인이 적용되는 노드를 삭제할 수는 있습니다. 예약 노드 할인이 적용되는 노드의 삭제 프로세스는 다른 노드를 삭제할 때와 동일합니다.

예약 노드 할인이 적용되는 노드를 삭제할 경우에는 다르지만 서로 사양이 호환되는 노드로 시작할 수 있습니다. 이 경우, 예약 기간(1년 또는 3년)에 요금 할인을 계속 받을 수 있습니다.

예약 노드 사용

AWS Management 콘솔, AWS Command Line Interface, 및 MemoryDB API를 사용하여 예약 노드 작업을 수행할 수 있습니다.

콘솔

사용 가능한 예약 노드 제품에 대한 가격 및 정보를 가져오려면

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 탐색 창에서 예약 노드를 선택합니다.
3. 예약 노드 구매를 선택합니다.
4. 노드 유형에서는 배포하려는 노드 유형을 선택합니다.
5. 수량에서 배포하려는 노드 수를 선택합니다.
6. [Term]에서 데이터베이스 노드를 예약할 기간을 선택합니다.
7. 제공 유형에서 해당 제공 유형을 선택합니다.

이렇게 선택하면 예약 요약에서 요금 정보가 표시됩니다.

> **Important**
>
> [Cancel]을 선택하면 이 예약 노드를 구입하지 않으며 요금이 발생하지 않습니다.

구매할 수 있는 노드 예약 상품에 대한 정보를 확인하였으면 이제 정보를 사용하여 다음 절차에 따라 상품을 구매할 수 있습니다.

예약 노드를 구매하려면

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 탐색 창에서 예약 노드를 선택합니다.
3. 예약 노드 구매를 선택합니다.

4. 노드 유형에서는 배포하려는 노드 유형을 선택합니다.
5. 수량에서 배포하려는 노드 수를 선택합니다.
6. [Term]에서 데이터베이스 노드를 예약할 기간을 선택합니다.
7. 제공 유형에서 해당 제공 유형을 선택합니다.
8. (선택 사항) - 예약 노드를 조회할 수 있도록 구매하는 예약 노드 자체 식별자를 할당할 수 있습니다. Reserved ID에 자신이 예약한 노드 식별자를 입력하면 됩니다.

   이렇게 선택하면 예약 요약에서 요금 정보가 표시됩니다.
9. 예약 노드 구매를 선택합니다.
10. 예약 노드를 구매하면 예약 노드 목록에 표시됩니다.

AWS 계정에 대한 예약 노드 관련 정보를 가져오려면

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 탐색 창에서 예약 노드를 선택합니다.
3. 현재 계정에서 예약한 노드가 나타납니다. 특정 예약 노드의 세부 정보를 보려면 목록에서 해당 노드를 선택합니다. 그러면 해당 노드에 대한 세부 정보가 표시됩니다.

AWS Command Line Interface

다음 `describe-reserved-nodes-offerings` 예에서는 예약 노드 오퍼링의 세부 정보를 반환합니다.

```
aws memorydb describe-reserved-nodes-offerings
```

그러면 다음과 비슷한 출력이 생성됩니다.

```
{
    "ReservedNodesOfferings": [
        {
            "ReservedNodesOfferingId": "0193cc9d-7037-4d49-b332-xxxxxxxxxxxx",
            "NodeType": "db.xxx.large",
            "Duration": 94608000,
            "FixedPrice": $xxx.xx,
```

```
                "OfferingType": "Partial Upfront",
                "RecurringCharges": [
                    {
                        "RecurringChargeAmount": $xx.xx,
                        "RecurringChargeFrequency": "Hourly"
                    }
                ]
            }
        ]
}
```

다음 파라미터를 전달하여 반환되는 범위를 제한할 수도 있습니다.

- `--reserved-nodes-offering-id` – 구매하려는 오퍼링의 ID입니다.
- `--node-type` - 노드 유형 필터 값입니다. 이 파라미터를 사용하면 지정된 노드 유형과 일치하는 예약만 표시할 수 있습니다.
- `--duration` - 년 또는 초 단위로 지정된 기간 필터 값입니다. 이 파라미터를 사용하면 해당 기간의 예약만 표시할 수 있습니다.
- `--offering-type` - 이 파라미터를 사용하면 지정된 개설과목 유형과 일치하는 사용 가능한 개설과목만 표시할 수 있습니다.

구매할 수 있는 노드 예약 상품에 대한 정보를 확인하였으면 이제 정보를 사용하여 오퍼링을 구매할 수 있습니다.

다음 `purchase-reserved-nodes-offering` 예에서는 새 예약 노드를 구매합니다.

Linux, macOS, Unix의 경우:

```
aws memorydb purchase-reserved-nodes-offering \

    --reserved-nodes-offering-id 0193cc9d-7037-4d49-b332-d5e984f1d8ca \
    --reservation-id reservation \
    --node-count 2
```

Windows의 경우:

```
aws memorydb purchase-reserved-nodes-offering ^
    --reserved-nodes-offering-id 0193cc9d-7037-4d49-b332-d5e984f1d8ca ^
    --reservation-id MyReservation
```

- `--reserved-nodes-offering-id`은(는) 구매를 제안하는 예약 노드의 이름을 나타냅니다.
- `--reservation-id`은(는) 이 예약을 추적하기 위한 고객 지정 식별자입니다.

> **Note**
>
> 예약 ID는 고객마다 고유한 식별자이며 이 예약을 추적하는 데 사용됩니다. 이 파라미터를 지정하지 않으면 MemoryDB에서 자동으로 예약 식별자를 생성합니다.

- `--node-count` 예약할 노드 수입니다. 기본값은 1입니다.

그러면 다음과 비슷한 출력이 생성됩니다.

```
{
    "ReservedNode": {
        "ReservationId": "reservation",
        "ReservedNodesOfferingId": "0193cc9d-7037-4d49-b332-xxxxxxxxxxxx",
        "NodeType": "db.xxx.large",
        "StartTime": 1671173133.982,
        "Duration": 94608000,
        "FixedPrice": $xxx.xx,
        "NodeCount": 2,
        "OfferingType": "Partial Upfront",
        "State": "payment-pending",
        "RecurringCharges": [
            {
                "RecurringChargeAmount": $xx.xx,
                "RecurringChargeFrequency": "Hourly"
            }
        ],
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxxx:reservednode/reservation"
    }
}
```

예약 노드를 구매한 후에는 예약 노드에 대한 정보를 가져올 수 있습니다.

다음 `describe-reserved-nodes` 예에서는 이 계정의 예약 노드에 대한 정보를 반환합니다.

```
aws memorydb describe-reserved-nodes
```

그러면 다음과 비슷한 출력이 생성됩니다.

```
{
    "ReservedNodes": [
        {
            "ReservationId": "ri-2022-12-16-00-28-40-600",
            "ReservedNodesOfferingId": "0193cc9d-7037-4d49-b332-xxxxxxxxxxxx",
            "NodeType": "db.xxx.large",
            "StartTime": 1671150737.969,
            "Duration": 94608000,
            "FixedPrice": $xxx.xx,
            "NodeCount": 1,
            "OfferingType": "Partial Upfront",
            "State": "active",
            "RecurringCharges": [
                {
                    "RecurringChargeAmount": $xx.xx,
                    "RecurringChargeFrequency": "Hourly"
                }
            ],
            "ARN": "arn:aws:memorydb:us-east-1:xxxxxxxx:reservednode/
ri-2022-12-16-00-28-40-600"
        }
    ]
}
```

다음 파라미터를 전달하여 반환되는 범위를 제한할 수도 있습니다.

- `--reservation-id` – 구매하는 예약 노드에 자체 식별자를 할당하여 추적할 수 있습니다.
- `--reserved-nodes-offering-id` - 오퍼링 식별자 필터 값. 이 파라미터를 사용하면 지정된 오퍼링 식별자와 일치하는 구매한 예약만 표시할 수 있습니다.
- `--node-type` - 노드 유형 필터 값입니다. 이 파라미터를 사용하면 지정된 노드 유형과 일치하는 예약만 표시할 수 있습니다.
- `--duration` - 년 또는 초 단위로 지정된 기간 필터 값입니다. 이 파라미터를 사용하면 해당 기간의 예약만 표시할 수 있습니다.
- `--offering-type` - 이 파라미터를 사용하면 지정된 개설과목 유형과 일치하는 사용 가능한 개설과목만 표시할 수 있습니다.

MemoryDB API

예약 노드를 통해 MemoryDB 쿼리 API를 사용하는 방법을 보여줍니다.

DescribeReservedNodesOfferings

예약 노드 오퍼링의 세부 정보를 반환합니다.

```
https://memorydb.us-west-2.amazonaws.com/
    ?Action=DescribeReservedNodesOfferings
    &ReservedNodesOfferingId=649fd0c8-xxxx-xxxx-xxxx-06xxxx75e95f
 &"Duration": 94608000,
    &NodeType="db.r6g.large"
    &OfferingType="Partial Upfront"
    &Version=2021-01-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

다음 파라미터는 반환되는 항목의 범위를 제한합니다.

- `ReservedNodesOfferingId` 구매를 제안하는 예약 노드의 이름을 나타냅니다.
- `Duration` - 년 또는 초 단위로 지정된 기간 필터 값입니다. 이 파라미터를 사용하면 해당 기간의 예약만 표시할 수 있습니다.
- `NodeType` - 노드 유형 필터 값입니다. 이 파라미터를 사용하면 지정된 노드 유형과 일치하는 오퍼링만 표시할 수 있습니다.
- `OfferingType` - 이 파라미터를 사용하면 지정된 개설과목 유형과 일치하는 사용 가능한 개설과목만 표시할 수 있습니다.

구매할 수 있는 노드 예약 상품에 대한 정보를 확인하였으면 이제 정보를 사용하여 오퍼링을 구매할 수 있습니다.

PurchaseReservedNodesOffering

예약 노드 상품을 구매할 수 있습니다.

```
https://memorydb.us-west-2.amazonaws.com/
    ?Action=PurchasedReservedNodesOffering
    &ReservedNodesOfferingId=649fd0c8-xxxx-xxxx-xxxx-06xxxx75e95f
    &ReservationID=myreservationID
    &NodeCount=1
    &Version=2021-01-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

- `ReservedNodesOfferingId` 구매를 제안하는 예약 노드의 이름을 나타냅니다.
- `ReservationID`은(는) 이 예약을 추적하기 위한 고객 지정 식별자입니다.

> **Note**
> 예약 ID는 고객마다 고유한 식별자이며 이 예약을 추적하는 데 사용됩니다. 이 파라미터를 지정하지 않으면 MemoryDB에서 자동으로 예약 식별자를 생성합니다.

- `NodeCount` 예약할 노드 수입니다. 기본값은 1입니다.

예약 노드를 구매한 후에는 예약 노드에 대한 정보를 가져올 수 있습니다.

DescribeReservedNodes

이 계정의 예약 노드에 대한 정보를 반환합니다.

```
https://memorydb.us-west-2.amazonaws.com/
 ?Action=DescribeReservedNodes
 &ReservedNodesOfferingId=649fd0c8-xxxx-xxxx-xxxx-06xxxx75e95f
 &ReservationID=myreservationID
 &NodeType="db.r6g.large"
 &Duration=94608000
 &OfferingType="Partial Upfront"
 &Version=2021-01-01
 &SignatureVersion=4
 &SignatureMethod=HmacSHA256
```

```
&Timestamp=20141201T220302Z
&X-Amz-Algorithm
&X-Amz-SignedHeaders=Host
&X-Amz-Expires=20141201T220302Z
&X-Amz-Credential=<credential>
&X-Amz-Signature=<signature>
```

다음 파라미터는 반환되는 항목의 범위를 제한합니다.

- `ReservedNodesOfferingId` 예약 노드의 이름을 나타냅니다.
- `ReservationID` – 구매하는 예약 노드에 자체 식별자를 할당하여 추적할 수 있습니다.
- `NodeType` - 노드 유형 필터 값입니다. 이 파라미터를 사용하면 지정된 노드 유형과 일치하는 예약만 표시할 수 있습니다.
- `Duration` - 년 또는 초 단위로 지정된 기간 필터 값입니다. 이 파라미터를 사용하면 해당 기간의 예약만 표시할 수 있습니다.
- `OfferingType` - 이 파라미터를 사용하면 지정된 개설과목 유형과 일치하는 사용 가능한 개설과목만 표시할 수 있습니다.

예약 노드에 대한 청구서 보기

예약 노드에 대한 결제는 AWS Management 콘솔의 결제 대시보드(Billing Dashboard)에서 확인할 수 있습니다.

예약 노드 결제 확인

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 콘솔 상단의 검색 버튼에서 결제를 선택합니다.
3. 대시보드 왼쪽에서 Bills를 선택합니다.
4. AWS서비스 요금에서 MemoryDB를 확장합니다.
5. 예약 노드가 있는 AWS리전을 확장하세요 (예: 미국 동부 (버지니아 북부)).

예약 노드 및 현재 월의 시간당 요금은 Amazon MemoryDB CreateCluster 예약 인스턴스에서 볼 수 있습니다.

Amazon MemoryDB CreateCluster Reserved Instances		[illegible]
AmazonMemoryDB, db.r6g.large reserved instance applied	81.000 Hrs	[illegible]
AmazonMemoryDB, db.r6g.4xlarge reserved instance applied	324.000 Hrs	[illegible]
AmazonMemoryDB, db.r6g.4xlarge reserved instance applied	162.000 Hrs	[illegible]
USD [illegible] hourly fee per AmazonMemoryDB, db.r6g.large instance	1,488.000 Hrs	[illegible]
USD [illegible] hourly fee per AmazonMemoryDB, db.r6gd.2xlarge instance	744.000 Hrs	[illegible]
USD 0[illegible] hourly fee per AmazonMemoryDB, db.r6g.4xlarge instance	744.000 Hrs	[illegible]
USD [illegible] hourly fee per AmazonMemoryDB, db.r6gd.xlarge instance	744.000 Hrs	[illegible]
USD [illegible] hourly fee per AmazonMemoryDB, db.r6gd.4xlarge instance	2,976.000 Hrs	[illegible]

노드 교체

MemoryDB는 패치 및 업그레이드를 통해 플릿을 일반적으로 원활하게 자주 업그레이드합니다. 하지만 기본 호스트에 필수 OS 업데이트를 적용하기 위해 MemoryDB 노드를 다시 시작해야 하는 경우가 있습니다. 보안, 신뢰성, 운영 성능을 강화하는 업그레이드 적용에 있어 이러한 교체가 필요합니다.

예정된 노드 교체 주기 이전에 언제든 이러한 교체를 직접 관리할 수 있는 옵션이 있습니다. 직접 대체를 관리할 때 노드를 다시 시작하면 인스턴스에서 OS 업데이트를 수신하고, 예정된 노드 대체는 취소됩니다. 노드 대체가 발생한다는 경고를 계속 수신할 수 있습니다. 이미 유지 관리의 필요성을 수동으로 완화한 경우 이 경고를 무시할 수 있습니다.

> **Note**
>
> MemoryDB에서 자동으로 생성된 교체 노드는 IP 주소가 다를 수 있습니다. 애플리케이션 구성을 검토하여 노드가 적절한 IP 주소와 연결되어 있는지 확인해야 합니다.

다음 목록은 MemoryDB에서 노드 하나의 교체를 예약할 경우 취할 수 있는 조치를 보여줍니다.

MemoryDB 노드 교체 옵션

- 아무 작업 안 함 - 아무 작업도 하지 않으면 MemoryDB가 예약대로 노드를 교체합니다.

  노드가 다중 AZ 클러스터의 구성원일 경우, MemoryDB는 패치 중 개선된 가용성과 업데이트, 기타 유지 관리 관련 노드 교체를 제공합니다.

  클러스터가 들어오는 쓰기 요청을 처리하는 동안 교체가 완료됩니다.

- 유지 관리 기간 변경 - 예약된 유지 관리 이벤트의 경우 MemoryDB에서 이메일 또는 알림 이벤트를 수신합니다. 이러한 경우 예약된 대체 시간 전에 유지 관리 기간을 변경하면 이제 노드가 새 시간에 대체됩니다. 자세한 내용은 MemoryDB 클러스터 수정 섹션을 참조하세요.

> **Note**
>
> 유지 관리 기간을 이동해 교체 기간을 변경하는 기능은 MemoryDB 알림에 유지 관리 기간이 포함된 경우에만 사용할 수 있습니다. 알림에 유지 관리 기간이 포함되어 있지 않으면 교체 기간을 변경할 수 없습니다.

예를 들어 11월 9일 목요일 15:00이고, 다음 유지 관리 기간은 11월 10일 금요일 17:00라고 가정해 보겠습니다. 다음은 이러한 가정의 결과를 보여주는 3가지 시나리오입니다.

- 유지 관리 기간을 현재 날짜/시간 이후이고 예약된 다음 유지 관리 기간 이전인 금요일 16:00으로 변경합니다. 11월 10일 금요일 16:00에 노드가 대체됩니다.
- 유지 관리 기간을 현재 날짜/시간 이후이고 예약된 다음 유지 관리 기간 이후인 토요일 16:00으로 변경합니다. 11월 11일 토요일 16:00에 노드가 대체됩니다.
- 유지 관리 기간을 현재 날짜/시간보다 일주일 빠른 수요일 오후 4시로 변경합니다. 11월 15일 수요일 16:00에 노드가 대체됩니다.

지침은 유지 관리 관리 중 단원을 참조하세요.

클러스터 관리

클러스터 수준에서 대부분의 MemoryDB 작업이 수행됩니다. 특정 수의 노드 및 각 노드에 대한 속성을 제어하는 파라미터 그룹을 사용하여 클러스터를 설정할 수 있습니다. 클러스터 하나에 속한 모든 노드는 노드 유형, 파라미터 및 보안 그룹 설정이 동일합니다.

클러스터마다 클러스터 식별자가 있습니다. 클러스터 식별자는 고객이 제공하는 클러스터 이름입니다. MemoryDB API 및 AWS CLI 명령과 상호 작용할 때 이 식별자가 특정한 클러스터를 지정합니다. 클러스터 식별자는 한 AWS 리전 내의 해당 고객에 대해 고유해야 합니다.

MemoryDB 클러스터는 Amazon EC2 인스턴스를 사용하여 액세스하도록 설계되었습니다. Amazon VPC 서비스 기반의 Virtual Private Cloud(VPC)에서만 MemoryDB 클러스터를 시작할 수 있으며 AWS 밖에서 액세스할 수 있습니다. 자세한 내용은 AWS 외부에서 MemoryDB 리소스에 액세스 섹션을 참조하세요.

데이터 계층화

r6gd 패밀리의 노드 유형을 사용하는 클러스터는 메모리와 로컬 SSD(solid state drives) 스토리지 간에 데이터를 계층화합니다. 데이터 계층화는 데이터를 메모리에 저장하는 것 외에도 각 클러스터 노드에서 저렴한 SSD(Solid State Drive)를 활용하여 Valkey 및 Redis OSS 워크로드에 대한 새로운 가격 대비 성능 옵션을 제공합니다. 다른 노드 유형과 마찬가지로 r6gd 노드에 기록된 데이터는 다중 AZ 트랜잭션 로그에 안정적으로 저장됩니다. 데이터 계층화는 전체 데이터 세트의 최대 20%까지 정기적으로 액세스하는 워크로드와 SSD에서 데이터에 액세스할 때 추가 지연 시간을 허용할 수 있는 애플리케이션에 이상적입니다.

데이터 계층화가 있는 클러스터에서 MemoryDB는 저장하는 모든 항목의 마지막 액세스 시간을 모니터링합니다. 사용 가능한 메모리(DRAM)가 모두 사용되면 MemoryDB는 최근최소사용(LRU) 알고리즘을 사용하여 자주 액세스하지 않는 항목을 메모리에서 SSD로 자동으로 이동합니다. 이후에 SSD의 데이터에 액세스하면 MemoryDB가 요청을 처리하기 전에 자동 및 비동기식으로 다시 메모리로 이동합니다. 데이터의 하위 집합에만 정기적으로 액세스하는 워크로드가 있는 경우 데이터 계층화는 용량을 비용 효율적으로 확장할 수 있는 최적의 방법입니다.

데이터 계층화를 사용하면 키 자체는 항상 메모리에 남아 있지만 LRU는 메모리 대 디스크의 값 배치를 제어합니다. 일반적으로 데이터 계층화를 사용하는 경우 키 크기가 값 크기보다 작은 것이 좋습니다.

데이터 계층화는 애플리케이션 워크로드에 미치는 성능 영향을 최소화하도록 설계되었습니다. 예를 들어 500바이트 문자열 값을 가정하면 SSD에 저장된 데이터에 대한 읽기 요청 및 메모리의 데이터에 대한 읽기 요청과 비교하여 평균 450마이크로초의 지연 시간을 추가로 기대할 수 있습니다.

가장 큰 데이터 계층화 노드 크기(db.r6gd.8xlarge)를 사용하면 단일 500노드 클러스터에 대략 500TB까지 저장할 수 있습니다(읽기 복제본 1개를 사용하는 경우는 250TB). 데이터 계층화의 경우, MemoryDB는 노드당(DRAM) 메모리의 19%를 비데이터 사용을 위해 예약합니다. 데이터 계층화는 MemoryDB에서 지원되는 모든 Valkey 및 Redis OSS 명령 및 데이터 구조와 호환됩니다. 이 기능을 사용하려면 클라이언트 측 변경 사항이 필요하지 않습니다.

주제

- 모범 사례
- 데이터 계층화 제한 사항
- 데이터 계층화 요금
- 데이터 계층화 모니터링
- 데이터 계층화 사용
- 스냅샷에서 클러스터로 데이터 복원

모범 사례

다음 모범 사례를 따르는 것이 좋습니다.

- 데이터 계층화는 전체 데이터 세트의 최대 20%까지 정기적으로 액세스하는 워크로드와 SSD에서 데이터에 액세스할 때 추가 지연 시간을 허용할 수 있는 애플리케이션에 이상적입니다.
- 데이터 계층화 노드에서 사용 가능한 SSD 용량을 사용하는 경우 값 크기가 키 크기보다 큰 것이 좋습니다. 값 크기는 128MB를 초과할 수 없습니다. 그렇지 않으면 디스크로 이동되지 않습니다. DRAM과 SSD 간에 항목이 이동하면 키는 항상 메모리에 남아 있고 값만 SSD 계층으로 이동합니다.

데이터 계층화 제한 사항

데이터 계층화에는 다음과 같은 제한 사항이 있습니다.

- 사용하는 노드 유형은 `us-east-2`, `us-east-1`, `us-west-2`, `us-west-1`, `eu-west-1`, `eu-west-3`, `eu-central-1`, `ap-northeast-1`, `ap-southeast-1`, `ap-southeast-2`, `ap-south-1`, `ca-central-1`, `sa-east-1`과 같은 리전에서 사용할 수 있는 r6gd 패밀리의 노드 유형이어야 합니다.
- r6gd 클러스터를 사용하지 않는 한 r6gd 클러스터의 스냅샷을 다른 클러스터로 복원할 수 없습니다.
- 데이터 계층화 클러스터를 위해 스냅샷을 Amazon S3로 내보낼 수 없습니다.
- Forkless 저장은 지원되지 않습니다.
- 데이터 계층화 클러스터(예: r6gd 노드 유형을 사용하는 클러스터)에서 데이터 계층화를 사용하지 않는 클러스터(예: r6g 노드 유형을 사용하는 클러스터)로 확장은 지원되지 않습니다.
- 데이터 계층화는 `volatile-lru`, `allkeys-lru`, `noeviction` 메모리 사용량 제한 정책만 지원합니다.
- 128MiB보다 큰 항목은 SSD로 이동되지 않습니다.

데이터 계층화 요금

R6gD 노드는 총 용량(메모리 + SSD)의 5배 더 많으며 R6g 노드에 비해 최대 사용률로 실행될 때 보관 비용의 60% 이상의 절감 효과를 얻을 수 있습니다(메모리만 해당). 자세한 내용은 MemoryDB 요금 단원을 참조하세요.

데이터 계층화 모니터링

MemoryDB는 데이터 계층화를 사용하는 성능 클러스터를 모니터링하도록 특별히 설계된 지표를 제공합니다. SSD와 비교하여 DRAM의 항목 비율을 모니터링하려면 MemoryDB에 대한 지표에서 `CurrItems` 지표를 사용할 수 있습니다. 백분율은 `(CurrItems with Dimension: Tier = Memory * 100) / (CurrItems with no dimension filter)`와(과) 같이 계산할 수 있습니다.

구성된 제거 정책이 허용하는 경우 메모리의 항목 비율이 5% 미만으로 감소하면 MemoryDB가 항목 제거를 시작합니다. 제거 정책으로 구성된 노드에서 쓰기 작업은 메모리 부족 오류를 수신합니다.

메모리에 있는 항목의 비율이 5% 미만으로 감소하면 여전히 MemoryDB 클러스터 크기 조정을 고려하는 것이 좋습니다. 자세한 내용은 MemoryDB에 대한 지표에서 데이터 계층화를 사용하는 MemoryDB 클러스터의 지표를 참조하세요.

데이터 계층화 사용

AWS Management 콘솔을 사용하여 데이터 계층화 사용

클러스터를 생성할 때 db.r6gd.xlarge와 같은 r6gd 패밀리의 노드 유형을 선택하여 데이터 계층화를 사용합니다. 해당 노드 유형을 선택하면 데이터 계층화가 자동으로 사용됩니다.

클러스터 생성에 대한 자세한 내용은 2단계: 클러스터 생성 섹션을 참조하세요.

AWS CLI를 사용하여 데이터 계층화 사용

AWS CLI를 사용하여 클러스터를 생성하는 경우, db.r6gd.xlarge와 같은 r6gd 패밀리에서 노드 유형을 선택하고 `--data-tiering` 파라미터를 설정하여 데이터 계층화를 사용합니다.

r6gd 패밀리의 노드 유형을 선택하는 경우 데이터 계층화를 선택 해제할 수 없습니다. `--no-data-tiering` 파라미터를 설정하는 경우 작업이 실패합니다.

Linux, macOS, Unix의 경우:

```
aws memorydb create-cluster \
   --cluster-name my-cluster \
   --node-type db.r6gd.xlarge \
   --engine valkey  \
   --acl-name my-acl \
   --subnet-group my-sg \
```

```
    --data-tiering
```

Windows의 경우:

```
aws memorydb create-cluster ^
    --cluster-name my-cluster ^
    --node-type db.r6gd.xlarge ^
    --engine valkey ^
    --acl-name my-acl ^
    --subnet-group my-sg
    --data-tiering
```

이 작업을 실행하면 다음과 유사한 응답이 표시됩니다.

```
{
    "Cluster": {
        "Name": "my-cluster",
        "Status": "creating",
        "NumberOfShards": 1,
        "AvailabilityMode": "MultiAZ",
        "ClusterEndpoint": {
            "Port": 6379
        },
        "NodeType": "db.r6gd.xlarge",
        "EngineVersion": "7.2",
        "EnginePatchVersion": "7.2.6",
        "Engine": "valkey"
        "ParameterGroupName": "default.memorydb-valkey7",
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxxxxxxxxx:cluster/my-cluster",
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
        "SnapshotWindow": "04:30-05:30",
        "ACLName": "my-acl",
        "DataTiering":"true",
        "AutoMinorVersionUpgrade": true
    }
}
```

스냅샷에서 클러스터로 데이터 복원

(콘솔), (AWS CLI) 또는 (MemoryDB API)를 사용하여 데이터 계층화가 활성화된 새 클러스터로 스냅샷을 복원할 수 있습니다. r6gd 패밀리의 노드 유형을 사용하여 클러스터를 생성하는 경우 데이터 계층화가 활성화됩니다.

데이터 계층화가 활성화된 상태로 스냅샷에서 클러스터로 데이터 복원(콘솔)

데이터 계층화가 활성화된 새 클러스터(콘솔)로 스냅샷을 복원하려면 스냅샷에서 복원(콘솔)의 단계를 따르세요.

데이터 계층화를 활성화하려면 r6gd 패밀리의 노드 유형을 선택해야 합니다.

데이터 계층화가 활성화된 상태에서 스냅샷에서 클러스터로 데이터 복원(AWS CLI)

AWS CLI를 사용하여 클러스터를 생성하는 경우, 데이터 계층화는 기본적으로 db.r6gd.xlarge와 같은 r6gd 패밀리의 노드 유형을 선택하고 `--data-tiering` 파라미터를 설정하여 사용됩니다.

r6gd 패밀리의 노드 유형을 선택하는 경우 데이터 계층화를 선택 해제할 수 없습니다. `--no-data-tiering` 파라미터를 설정하는 경우 작업이 실패합니다.

Linux, macOS, Unix의 경우:

```
aws memorydb create-cluster \
   --cluster-name my-cluster \
   --node-type db.r6gd.xlarge \
   --engine valkey
   --acl-name my-acl \
   --subnet-group my-sg \
   --data-tiering \
   --snapshot-name my-snapshot
```

Windows의 경우:

```
aws memorydb create-cluster ^
   --cluster-name my-cluster ^
   --node-type db.r6gd.xlarge ^
   --engine valkey ^
   --acl-name my-acl ^
   --subnet-group my-sg ^
   --data-tiering ^
   --snapshot-name my-snapshot
```

이 작업을 실행하면 다음과 유사한 응답이 표시됩니다.

```
{
    "Cluster": {
        "Name": "my-cluster",
        "Status": "creating",
        "NumberOfShards": 1,
        "AvailabilityMode": "MultiAZ",
        "ClusterEndpoint": {
            "Port": 6379
        },
        "NodeType": "db.r6gd.xlarge",
        "EngineVersion": "7.2",
        "EnginePatchVersion": "7.2.6",
        "Engine": "valkey"
        "ParameterGroupName": "default.memorydb-valkey7",
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxxxxxxxxx:cluster/my-cluster",
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
        "SnapshotWindow": "04:30-05:30",
        "ACLName": "my-acl",
        "DataTiering": "true"
}
```

클러스터 준비

다음에 MemoryDB 콘솔, AWS CLI 또는 MemoryDB API를 사용하는 클러스터 생성에 관한 지침이 나와 있습니다.

클러스터를 생성할 때마다 준비 작업을 미리 하면 즉시 업그레이드하거나 변경할 필요가 없어 좋습니다.

주제

- 요구 사항 결정

요구 사항 결정

준비

다음 질문에 대한 답을 알고 있으면 클러스터를 더 원활하게 만들 수 있습니다.

- 클러스터를 생성하기 전에 동일한 VPC에서 서브넷 그룹을 생성해야 합니다. 또는 제공된 기본 서브넷 그룹을 사용할 수도 있습니다. 자세한 내용은 서브넷 및 서브넷 그룹 섹션을 참조하세요.

  MemoryDB는 Amazon EC2를 사용하여 AWS 내에서 액세스하도록 고안되었습니다. 하지만 Amazon VPC 기반의 VPC에서 시작하는 경우 AWS 밖에서 액세스 권한을 제공할 수 있습니다. 자세한 내용은 AWS 외부에서 MemoryDB 리소스에 액세스 섹션을 참조하세요.
- 파라미터 값을 사용자 지정해야 합니까?

  그렇다면 사용자 지정 파라미터 그룹을 만듭니다. 자세한 내용은 파라미터 그룹 생성 섹션을 참조하세요.
- VPC 보안 그룹을 생성해야 합니까?

  자세한 내용은 VPC의 보안을 참조하세요.
- 어떤 방법으로 내결함성을 구현하시겠습니까?

  자세한 내용은 장애 완화 섹션을 참조하세요.

주제

메모리 및 프로세서 요구 사항

MemoryDB의 기본 구성 요소는 노드입니다. 노드는 샤드로 구성되어 클러스터를 형성합니다. 클러스터에 사용할 노드 유형을 결정할 때 클러스터의 노드 구성과 저장해야 하는 데이터의 양을 고려합니다.

MemoryDB 클러스터 구성

MemoryDB 클러스터는 1개에서 500개의 샤드로 구성됩니다. MemoryDB 클러스터에 있는 데이터는 클러스터의 여러 샤드에 두루 분할됩니다. 애플리케이션은 엔드포인트라는 네트워크 주소를 사용하여 MemoryDB 클러스터에 연결됩니다. 노드 엔드포인트 외에도 MemoryDB 클러스터는 클러스터 엔드포

인트라는 엔드포인트를 가지고 있습니다. 애플리케이션에서는 이 엔드포인트를 사용하여 클러스터에서 읽거나 쓸 수 있으며, 읽을 노드 또는 쓸 노드에 대한 결정은 MemoryDB에 맡깁니다.

향상된 I/O 멀티플렉싱

Valkey 또는 Redis OSS 버전 7.0 이상을 실행하는 경우 향상된 I/O 멀티플렉싱을 통해 추가 가속화를 얻을 수 있습니다. 각 전용 네트워크 IO는 여러 클라이언트의 명령을 엔진으로 파이프라인하며 명령을 배치로 효율성 있게 처리하는 기능을 활용합니다. 자세한 내용은 초고속 성능 및 the section called “지원되는 노드 유형”을(를) 참조하세요.

조정 요구 사항

모든 클러스터를 더 큰 노드 유형으로 스케일 업할 수 있습니다. MemoryDB 클러스터를 스케일 업할 때 클러스터를 계속 사용할 수 있도록 온라인으로 조정하거나 스냅샷에서 새 클러스터를 확장하고 새 클러스터가 비워지지 않도록 할 수 있습니다.

자세한 내용은 이 가이드의 스케일링 을 참조하세요.

액세스 요구 사항

설계에 따라 MemoryDB 클러스터는 Amazon EC2 인스턴스에서 액세스합니다. MemoryDB 클러스터에 대한 네트워크 액세스는 클러스터를 생성한 계정으로 제한됩니다. 따라서 Amazon EC2 인스턴스에서 클러스터에 액세스하려면 먼저 클러스터에 액세스하도록 승인해야 합니다. 자세한 지침은 이 가이드의 3단계: 클러스터에 대한 액세스 허가를 참조하세요.

리전 및 가용 영역

애플리케이션과 가까운 AWS 리전에 MemoryDB 클러스터를 배치하면 지연 시간을 줄일 수 있습니다. 클러스터에 다중 노드가 있는 경우 다른 가용 영역에 노드를 배치하면 클러스터에 장애가 미치는 영향을 줄일 수 있습니다.

자세한 내용은 다음 자료를 참조하세요.

- 리전 및 가용 영역 선택
- 장애 완화

클러스터 생성

MemoryDB는 클러스터를 생성하는 세 가지 방법을 제공합니다. 자세한 내용은 2단계: 클러스터 생성 섹션을 참조하세요.

클러스터 세부 정보 보기

MemoryDB 콘솔, AWS CLI, 또는 MemoryDB API를 사용하여 클러스터 하나 이상에 대한 세부 정보를 볼 수 있습니다.

MemoryDB 클러스터의 세부 정보 보기(콘솔)

다음 절차에서는 MemoryDB 콘솔을 사용하여 MemoryDB 클러스터의 세부 정보를 보는 방법을 자세히 설명합니다.

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 클러스터의 세부 정보를 보려면 클러스터 이름의 왼쪽에 있는 라디오 버튼을 선택한 다음 세부 정보 보기를 선택합니다. 클러스터를 직접 클릭하여 클러스터 세부 정보 페이지를 볼 수도 있습니다.

   클러스터 세부 정보 페이지에는 클러스터 엔드포인트를 포함하여 클러스터에 대한 세부 정보가 표시됩니다. 클러스터 세부 정보 페이지에 있는 여러 탭을 사용하여 자세한 내용을 볼 수 있습니다.
3. 클러스터의 샤드 목록과 각 샤드 내 노드 개수를 보려면 샤드 및 노드 탭을 선택합니다.
4. 노드에 대한 특정 정보를 보려면 아래 표에서 샤드를 확장하세요. 또는 검색 상자를 사용하여 샤드를 검색할 수도 있습니다.

   이렇게 하면 가용 영역, 슬롯/키스페이스, 상태 등 각 노드에 대한 정보가 표시됩니다.
5. 지표 탭을 선택하여 CPU 사용률 및 엔진 CPU 사용률과 같은 각각의 프로세스를 모니터링할 수 있습니다. 자세한 내용은 MemoryDB에 대한 지표 섹션을 참조하세요.
6. 네트워크 및 보안 탭을 선택하여 서브넷 그룹 및 보안 그룹의 세부 정보를 확인합니다.
   a. 서브넷 그룹에서 서브넷 그룹의 이름, 서브넷이 속한 VPC로 연결되는 링크, 서브넷 그룹의 Amazon 리소스 이름(ARN)을 볼 수 있습니다.
   b. 보안 그룹에서는 보안 그룹 ID, 이름 및 설명을 볼 수 있습니다.
7. 유지 관리 및 스냅샷 탭을 선택하면 스냅샷 설정의 세부 정보를 볼 수 있습니다.
   a. 스냅샷에서 자동 스냅샷의 활성화 여부, 스냅샷 보존 기간 및 스냅샷 창을 확인할 수 있습니다.
   b. 스냅샷에는 스냅샷 이름, 크기, 샤드 수 및 상태를 포함하여 이 클러스터에 대한 모든 스냅샷 목록이 표시됩니다.

자세한 내용은 스냅샷 및 복원 섹션을 참조하세요.

8. 유지 관리 및 스냅샷 탭을 선택하면 보류 중인 ACL, 리샤딩 또는 서비스 업데이트와 함께 유지 관리 기간의 세부 정보를 볼 수 있습니다. 자세한 내용은 유지 관리 관리 중 섹션을 참조하세요.
9. 서비스 업데이트 탭을 선택하면 이 클러스터에 적용되는 모든 서비스 업데이트의 세부 정보를 볼 수 있습니다. 자세한 내용은 MemoryDB의 서비스 업데이트 섹션을 참조하세요.
10. 태그 탭을 선택하면 이 클러스터와 관련된 리소스 또는 비용 할당 태그의 세부 정보를 볼 수 있습니다. 자세한 내용은 스냅샷 태깅 섹션을 참조하세요.

클러스터 세부 정보 보기(AWS CLI)

AWS CLI `describe-clusters` 명령을 사용하여 클러스터의 세부 정보를 볼 수 있습니다. `--cluster-name` 파라미터가 생략되면 여러 클러스터(최대 `--max-results`개)의 세부 정보가 반환됩니다. `--cluster-name` 파라미터가 포함되면 지정한 클러스터의 세부 정보가 반환됩니다. `--max-results` 파라미터를 사용하여 반환되는 레코드 수를 제한할 수 있습니다.

다음 코드는 `my-cluster`의 세부 정보를 나열합니다.

```
aws memorydb describe-clusters --cluster-name my-cluster
```

다음 코드는 클러스터 최대 25개의 세부 정보를 나열합니다.

```
aws memorydb describe-clusters --max-results 25
```

Example

Linux, macOS, Unix의 경우:

```
aws memorydb describe-clusters \
    --cluster-name my-cluster \
    --show-shard-details
```

Windows의 경우:

```
aws memorydb describe-clusters ^
    --cluster-name my-cluster ^
    --show-shard-details
```

다음 JSON 출력은 응답을 보여줍니다.


```
{
    "Clusters": [
        {
            "Name": "my-cluster",
            "Description": "my cluster",
            "Status": "available",
            "NumberOfShards": 1,
            "Shards": [
                {
                    "Name": "0001",
                    "Status": "available",
                    "Slots": "0-16383",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0001-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": 1629230643.961,
                            "Endpoint": {
                                "Address": "my-cluster-0001-001.my-
cluster.abcdef.memorydb.us-east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0001-002",
                            "Status": "available",
                            "CreateTime": 1629230644.025,
                            "Endpoint": {
                                "Address": "my-cluster-0001-002.my-
cluster.abcdef.memorydb.us-east-1.amazonaws.com",
                                "Port": 6379
                            }
                        }
                    ],
                    "NumberOfNodes": 2
                }
            ],
            "ClusterEndpoint": {
                "Address": "clustercfg.my-cluster.abcdef.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
```

```
            },
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "default",
            "TLSEnabled": true,
            "ARN": "arn:aws:memorydb:us-east-1:000000000:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
            "MaintenanceWindow": "sat:06:30-sat:07:30",
            "SnapshotWindow": "04:00-05:00",
            "ACLName": "open-access",
            "DataTiering": "false",
            "AutoMinorVersionUpgrade": true,
        }
```

자세한 내용은 AWS CLI에서 MemoryDB topic `describe-clusters` 을 참조하세요.

클러스터 세부 정보 보기(MemoryDB API)

MemoryDB API `DescribeClusters` 작업을 사용하여 클러스터의 세부 정보를 볼 수 있습니다. `ClusterName` 파라미터가 포함되면 지정한 클러스터의 세부 정보가 반환됩니다. `ClusterName` 파라미터가 생략되면 클러스터 최대 `MaxResults`개(기본값 100)의 세부 정보가 반환됩니다. `MaxResults`의 값은 20 이상 또는 100 이하여야 합니다.

다음 코드는 `my-cluster`의 세부 정보를 나열합니다.

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeClusters
   &ClusterName=my-cluster
   &Version=2021-01-01
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20210802T192317Z
   &X-Amz-Credential=<credential>
```

다음 코드는 클러스터 최대 25개의 세부 정보를 나열합니다.

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeClusters
```

```
&MaxResults=25
&Version=2021-02-02
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20210802T192317Z
&X-Amz-Credential=<credential>
```

자세한 내용은 MemoryDB API 참조 항목 `DescribeClusters`를 참조하세요.

MemoryDB 클러스터 수정

클러스터에서 노드를 추가하거나 제거하는 것 외에도 보안 그룹을 추가하거나 유지 관리 기간 또는 파라미터 그룹을 변경하는 등 기존의 클러스터를 변경해야 할 경우도 있습니다.

유지 관리 기간을 사용률이 가장 낮은 시간으로 낮추는 것이 유익하므로 수정해야 할 때도 있습니다.

클러스터의 파라미터를 변경하면 변경 사항이 클러스터에 즉시 적용됩니다. 이는 클러스터의 파라미터 그룹 자체에서 변경하든 파라미터 값을 클러스터의 파라미터 그룹 내에서 변경하든 마찬가지입니다.

클러스터의 엔진 버전을 업데이트할 수도 있습니다. 예를 들어 새 엔진 마이너 버전을 선택하면 MemoryDB가 즉시 클러스터 업데이트를 시작합니다.

AWS Management 콘솔 사용

클러스터를 수정하려면

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 상단 오른쪽 모서리의 목록에서 수정하려는 클러스터가 있는 AWS 리전을 선택합니다.
3. 왼쪽 탐색창에서 클러스터로 이동합니다. 클러스터 세부 정보에서 라디오 버튼을 사용하여 클러스터를 선택하고 작업, 수정으로 이동합니다.
4. 수정 페이지가 나타납니다.
5. Modify 창에서 원하는 내용을 수정합니다. 옵션에는 다음이 포함됩니다.

   - 설명
   - 서브넷 그룹
   - VPC 보안 그룹
   - 노드 유형

   > **Note**
   >
   > 클러스터가 r6gd 패밀리의 노드 유형을 사용하는 경우 해당 패밀리 내에서 다른 크기의 노드만 선택할 수 있습니다. r6gd 패밀리의 노드 유형을 선택하는 경우 데이터 계층화가 자동으로 활성화됩니다. 자세한 내용은 데이터 계층화 섹션을 참조하세요.

   - Valkey 또는 Redis OSS 버전 호환성

- 자동 스냅샷 활성화
- 수동 스냅샷 보존 기간
- 스냅샷 창
- 유지 관리 기간
- SNS 알림에 대한 주제

6. 변경 사항 저장을 선택합니다.

클러스터 세부 정보 페이지로 이동하여 수정을 클릭하여 클러스터를 수정할 수도 있습니다. 클러스터의 특정 섹션을 수정하려면 클러스터 세부 정보 페이지의 해당 탭으로 이동하여 수정을 클릭하면 됩니다.

AWS CLI 사용

AWS CLI `update-cluster` 작업을 사용하여 기존의 클러스터를 수정할 수 있습니다. 클러스터의 구성 값을 수정하려면 클러스터 ID, 변경할 파라미터 및 파라미터의 새 값을 지정합니다. 다음 예제에서는 `my-cluster`라는 클러스터의 유지 관리 기간을 변경하고 변경 사항을 즉시 적용합니다.

Linux, macOS, Unix의 경우:

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --preferred-maintenance-window sun:23:00-mon:02:00
```

Windows의 경우:

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --preferred-maintenance-window sun:23:00-mon:02:00
```

자세한 내용은 AWS CLI 명령 레퍼런스의 update-cluster를 참조하세요.

MemoryDB API 사용

MemoryDB API UpdateCluster 작업을 사용하여 기존의 클러스터를 수정할 수 있습니다. 클러스터의 구성 값을 수정하려면 클러스터 ID, 변경할 파라미터 및 파라미터의 새 값을 지정합니다. 다음 예제에서는 `my-cluster`라는 클러스터의 유지 관리 기간을 변경하고 변경 사항을 즉시 적용합니다.

```
https://memory-db.us-east-1.amazonaws.com/
```

```
    ?Action=UpdateCluster
    &ClusterName=my-cluster
    &PreferredMaintenanceWindow=sun:23:00-mon:02:00
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210801T220302Z
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210802T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

Redis OSS에서 Valkey로의 엔진 간 업그레이드를 트리거하는 방법

콘솔, API 또는 CLI를 사용하여 기존 Redis OSS 클러스터를 Valkey 엔진으로 업그레이드할 수 있습니다.

기본 파라미터 그룹을 사용하는 기존 Redis OSS 클러스터가 있는 경우 update-cluster API를 사용하여 새 엔진 및 엔진 버전을 지정하여 Valkey로 업그레이드할 수 있습니다.

Linux, macOS, Unix의 경우:

```
aws memorydb update-cluster \
   --cluster-name myCluster \
   --engine valkey \
   --engine-version 7.2
```

Windows의 경우:

```
aws memorydb update-cluster ^
   --cluster-name myCluster ^
   --engine valkey ^
   --engine-version 7.2
```

업그레이드하려는 기존 Redis OSS 클러스터에 사용자 지정 파라미터 그룹이 적용된 경우 요청에서 사용자 지정 Valkey 파라미터 그룹도 전달해야 합니다. 입력 Valkey 사용자 지정 파라미터 그룹은 기존 Redis OSS 사용자 지정 파라미터 그룹과 동일한 Redis OSS 정적 파라미터 값을 가져야 합니다.

Linux, macOS, Unix의 경우:

```
aws memorydb update-cluster \
```

```
    --cluster-name myCluster \
    --engine valkey \
    --engine-version 7.2 \
    --parameter-group-name myParamGroup
```

Windows의 경우:

```
aws memorydb update-cluster ^
    --cluster-name myCluster ^
    --engine valkey ^
    --engine-version 7.2 ^
    --parameter-group-name myParamGroup
```

클러스터에서 노드 추가/제거

AWS Management 콘솔, AWS CLI 또는 MemoryDB API를 사용하여 클러스터에서 노드를 추가하거나 제거할 수 있습니다.

AWS Management 콘솔 사용

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 클러스터 목록에서 노드를 추가하거나 제거할 클러스터 이름을 선택합니다.
3. 샤드 및 노드 탭에서 노드 추가/삭제를 선택합니다.
4. 새로운 노드 수에 원하는 노드의 수를 입력합니다.
5. 확인을 선택합니다.

> **Important**
>
> 노드 수를 1로 설정하면 더 이상 다중 AZ를 사용할 수 없습니다. 자동 장애 조치를 활성화하도록 선택할 수도 있습니다.

AWS CLI 사용

1. 제거할 노드의 이름을 확인합니다. 자세한 내용은 클러스터 세부 정보 보기 섹션을 참조하세요.
2. 다음 예제와 같이 제거할 노드 목록과 함께 `update-cluster` CLI 작업을 사용하세요.

   명령줄 인터페이스를 사용하여 클러스터에서 노드를 제거하려면 다음 파라미터와 함께 `update-cluster` 명령을 사용하세요.

   - `--cluster-name` 노드를 제거할 클러스터의 ID입니다.
   - `--replica-configuration` - 복제본 수를 설정할 수 있습니다.
     - `ReplicaCount`- 속성을 설정하여 원하는 복제본 노드의 수를 지정합니다.
   - `--region` 노드를 제거할 클러스터의 AWS 리전을 지정합니다.

   Linux, macOS, Unix의 경우:

```
aws memorydb update-cluster \
```

```
    --cluster-name my-cluster \
    --replica-configuration \
        ReplicaCount=1 \
    --region us-east-1
```

Windows의 경우:

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --replica-configuration ^
        ReplicaCount=1 ^
    --region us-east-1
```

자세한 내용은 AWS CLI 항목 `update-cluster`를 참조하세요.

MemoryDB API 사용

MemoryDB API를 사용하여 노드를 제거하려면 다음과 같이 클러스터 이름 및 제거할 노드 목록과 함께 `UpdateCluster` API 작업을 직접적으로 호출하세요.

- `ClusterName` 노드를 제거할 클러스터의 ID입니다.
- `ReplicaConfiguration` - 복제본 수를 설정할 수 있습니다.
  - `ReplicaCount` - 속성을 설정하여 원하는 복제본 노드의 수를 지정합니다.
- `Region` 노드를 제거할 클러스터의 AWS 리전을 지정합니다.

자세한 내용은 UpdateCluster를 참조하세요.

클러스터 액세스

MemoryDB 인스턴스는 Amazon EC2 인스턴스를 통해 액세스하도록 설계되었습니다.

동일한 Amazon VPC의 Amazon EC2 인스턴스에서 MemoryDB 노드에 액세스할 수 있습니다. 또는 VPC 피어링을 사용하여 다른 Amazon VPC의 Amazon EC2에서 MemoryDB 노드에 액세스할 수 있습니다.

주제

- 클러스터에 액세스 권한 부여
- AWS 외부에서 MemoryDB 리소스에 액세스

클러스터에 액세스 권한 부여

동일한 Amazon VPC에서 실행 중인 Amazon EC2 인스턴스에서만 MemoryDB 클러스터에 연결할 수 있습니다. 이 경우 클러스터에 네트워크 진입을 허용해야 합니다.

Amazon VPC 보안 그룹에서 클러스터로의 네트워크 진입을 허용하려면

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 왼쪽 탐색 창의 Network & Security 아래에서 Security Groups를 선택합니다.
3. 보안 그룹 목록에서 Amazon VPC를 위한 보안 그룹을 선택합니다. MemoryDB 사용을 위한 보안 그룹을 생성하지 않는 한 이 보안 그룹의 이름은 default로 지정됩니다.
4. [Inbound] 탭을 선택하고 다음을 수행합니다.

    a. 편집을 선택합니다.

    b. 규칙 추가를 선택합니다.

    c. [Type] 열에서 [Custom TCP rule]을 선택합니다.

    d. [Port range] 상자에 클러스터 노드의 포트 번호를 입력합니다. 이 번호는 클러스터를 시작할 때 지정한 번호와 동일해야 합니다. Valkey와 Redis OSS의 기본 포트는 **6379**입니다.

    e. 소스 상자에서 포트 범위(0.0.0.0/0)를 가진 위치 무관을 선택하면 Amazon VPC 내에서 시작한 Amazon EC2 인스턴스를 MemoryDB 노드에 연결할 수 있습니다.

> **⚠ Important**
>
> MemoryDB 클러스터를 0.0.0.0/0으로 열면 공용 IP 주소가 없기 때문에 클러스터가 인터넷에 노출되지 않으므로 VPC 외부에서 액세스할 수 없습니다. 그러나 기본 보안 그룹이 고객 계정의 다른 Amazon EC2 인스턴스에 적용될 수 있으며 이러한 인스턴스는 공용 IP 주소를 가질 수 있습니다. 기본 포트에서 무언가를 실행하면 비의도적으로 해당 서비스가 노출될 수 있습니다. 따라서 MemoryDB가 독점적으로 사용하는 VPC 보안 그룹을 생성하는 것이 좋습니다. 자세한 정보는 사용자 지정 보안 그룹을 참조하세요.

f. 저장을 선택합니다.

Amazon EC2 인스턴스를 Amazon VPC로 시작하면 해당 인스턴스를 MemoryDB 클러스터에 연결할 수 있습니다.

AWS 외부에서 MemoryDB 리소스에 액세스

MemoryDB는 VPC에 내부적으로 사용하도록 설계된 서비스입니다. 인터넷 트래픽의 지연 시간 및 보안 문제로 인해 외부 액세스는 권장되지 않습니다. 그러나 테스트 또는 개발 목적으로 MemoryDB에 대한 외부 액세스가 필요한 경우, VPN을 통해 수행할 수 있습니다.

AWS Client VPN을 사용하면 다음과 같은 이점과 함께 MemoryDB 노드에 대한 외부 액세스를 허용합니다.

- 승인된 사용자 또는 인증 키에 대한 제한된 액세스
- VPN 클라이언트와 AWS VPN 엔드포인트 간의 암호화된 트래픽
- 특정 서브넷 또는 노드에 대한 제한된 액세스
- 사용자 또는 인증 키로부터의 액세스를 쉽게 취소
- 감사 연결

다음 절차에서는 다음 작업을 수행하는 방법을 보여줍니다.

주제

인증 기관 생성

다양한 기술이나 도구를 사용하여 CA(인증 기관)를 생성할 수 있습니다. OpenVPN 프로젝트에서 제공하는 easy-rsa 유틸리티를 사용하는 것이 좋습니다. 선택한 옵션에 관계없이 키를 안전하게 유지해야 합니다. 다음 절차에서는 easy-rsa 스크립트를 다운로드하고 인증 기관과 첫 번째 VPN 클라이언트를 인증하는 키를 생성합니다.

- 초기 인증서를 생성하려면 터미널을 열고 다음 작업을 수행하세요.
  - `git clone` https://github.com/OpenVPN/easy-rsa
  - `cd easy-rsa`
  - `./easyrsa3/easyrsa init-pki`
  - `./easyrsa3/easyrsa build-ca nopass`
  - `./easyrsa3/easyrsa build-server-full server nopass`

- `./easyrsa3/easyrsa build-client-full client1.domain.tld nopass`

  인증서를 포함하는 pki 하위 디렉터리는 easy-rsa 아래에 생성됩니다.

- 다음과 같이 AWS Certificate Manager(ACM)에 서버 인증서를 제출합니다.
  - ACM 콘솔에서 Certificate Manager를 선택합니다.
  - 인증서 가져오기를 선택합니다.
  - `easy-rsa/pki/issued/server.crt` 파일에서 사용할 수 있는 퍼블릭 키 인증서를 인증서 본문 필드에 입력합니다.
  - `easy-rsa/pki/private/server.key`에서 사용할 수 있는 프라이빗 키를 인증서 프라이빗 키 필드에 붙여 넣습니다. `BEGIN AND END PRIVATE KEY` 사이의 모든 선(`BEGIN` 및 `END` 선 포함)을 선택해야 합니다.
  - `easy-rsa/pki/ca.crt` 파일에서 사용할 수 있는 CA 퍼블릭 키를 인증서 체인 필드에 붙여넣습니다.
  - 검토 및 가져오기를 선택합니다.
  - 가져오기를 선택합니다.

  AWS CLI를 사용하여 서버의 인증서를 ACM에 제출하려면 다음 명령을 실행하세요. `aws acm import-certificate --certificate fileb://easy-rsa/pki/issued/server.crt --private-key file://easy-rsa/pki/private/server.key --certificate-chain file://easy-rsa/pki/ca.crt --region *region*`

  나중에 사용할 수 있도록 인증서 ARN을 기록해 둡니다.

AWS Client VPN 구성 요소 구성

AWS 콘솔 사용

AWS 콘솔에서 서비스, VPC 순으로 선택합니다.

Virtual Private Network(가상 프라이빗 네트워크)에서 Client VPN Endpoints(클라이언트 VPN 엔드포인트)를 선택하고 다음을 수행합니다.

AWS Client VPN 구성 요소 구성

- Client VPN 엔드포인트 생성을 선택합니다.
- 다음과 같은 옵션을 지정할 수 있습니다.

  - Client IPv4 CIDR(클라이언트 IPv4 CIDR): 넷마스크가 최소 /22 범위에 있는 프라이빗 네트워크를 사용합니다. 선택한 서브넷이 VPC 네트워크의 주소와 충돌하지 않는지 확인합니다. 예: 10.0.0.0/22.
  - Server certificate ARN(서버 인증서 ARN)에서 앞서 가져온 인증서의 ARN을 선택합니다.
  - Use mutual authentication(상호 인증 사용)을 선택합니다.
  - Client certificate ARN(클라이언트 인증서 ARN)에서 앞서 가져온 인증서의 ARN을 선택합니다.
  - Client VPN 엔드포인트 생성을 선택합니다.

사용AWS CLI

다음 명령을 실행합니다.

```
aws ec2 create-client-vpn-endpoint --client-cidr-block
"10.0.0.0/22" --server-certificate-arn arn:aws:acm:us-
east-1:012345678912:certificate/0123abcd-ab12-01a0-123a-123456abcdef --
authentication-options Type=certificate-
authentication,,MutualAuthentication={ClientRootCertificateChainArn=arn:aws:acm:
east-1:012345678912:certificate/123abcd-ab12-01a0-123a-123456abcdef} --
connection-log-options Enabled=false
```

출력 예시:

```
"ClientVpnEndpointId": "cvpn-endpoint-0123456789abcdefg",
"Status": { "Code": "pending-associate" }, "DnsName": "cvpn-
endpoint-0123456789abcdefg.prod.clientvpn.us-east-1.amazonaws.com" }
```

대상 네트워크를 VPN 엔드포인트에 연결

- 새 VPN 엔드포인트를 선택한 다음 연결 탭을 선택합니다.
- 연결을 선택하고 다음 옵션을 지정합니다.
  - VPC: MemoryDB 클러스터의 VPC를 선택합니다.
  - MemoryDB 클러스터의 네트워크 중 하나를 선택합니다. 의심스러운 경우, MemoryDB 대시보드의 서브넷 그룹에 있는 네트워크를 검토하세요.
  - 연결을 선택합니다. 필요한 경우 나머지 네트워크에 대해 이 단계를 반복합니다.

사용AWS CLI

다음 명령을 실행합니다.

```
aws ec2 associate-client-vpn-target-network --client-vpn-endpoint-id cvpn-
endpoint-0123456789abcdefg --subnet-id subnet-0123456789abdcdef
```

출력 예시:

```
"Status": { "Code": "associating" }, "AssociationId": "cvpn-
assoc-0123456789abdcdef" }
```

VPN 보안 그룹 검토

VPN 엔드포인트는 VPC의 기본 보안 그룹을 자동으로 채택합니다. 인바운드 및 아웃바운드 규칙을 점검하고 보안 그룹이 VPN 네트워크(VPN 엔드포인트 설정에 정의됨)에서 서비스 포트의 MemoryDB 네트워크로의 트래픽을 허용하는지 확인합니다(기본적으로 Redis의 경우, 6379).

VPN 엔드포인트에 할당된 보안 그룹을 변경해야 하는 경우 다음과 같이 진행합니다.

- 현재 보안 그룹을 선택합니다.
- Apply Security Group(보안 그룹 적용)을 선택합니다.
- 보안 그룹을 선택합니다.

사용AWS CLI

다음 명령을 실행합니다.

```
aws ec2 apply-security-groups-to-client-vpn-target-network --
client-vpn-endpoint-id cvpn-endpoint-0123456789abcdefga  --vpc-id
vpc-0123456789abdcdef --security-group-ids sg-0123456789abdcdef
```

출력 예시:

```
"SecurityGroupIds": [ "sg-0123456789abdcdef" ] }
```

> **Note**
> MemoryDB 보안 그룹은 VPN 클라이언트에서 오는 트래픽을 허용해야 합니다. 클라이언트의 주소는 VPC 네트워크에 따라 VPN 엔드포인트 주소로 마스킹 처리됩니다. 따라서 MemoryDB 보안 그룹에서 인바운드 규칙을 생성할 때 VPC 네트워크(VPN 클라이언트 네트워크가 아님)를 고려하세요.

대상 네트워크에 대한 VPN 액세스 승인

Authorization(권한 부여) 탭에서 Authorize Ingress(권한 부여 승인)를 선택하고 다음과 같이 지정합니다.

- 액세스를 활성화할 대상 네트워크: 0.0.0.0/0을 사용하여 모든 네트워크(인터넷 포함)에 대한 액세스를 허용하거나 MemoryDB 네트워크/호스트를 제한합니다.
- 다음에 대한 액세스 권한 부여:에서 모든 사용자에게 액세스 허용를 선택합니다.
- 권한 부여 규칙 추가를 선택합니다.

사용AWS CLI

다음 명령을 실행합니다.

```
aws ec2 authorize-client-vpn-ingress --client-vpn-endpoint-id cvpn-
endpoint-0123456789abcdefg --target-network-cidr 0.0.0.0/0 --authorize-all-
groups
```

출력 예시:

```
{ "Status": { "Code": "authorizing" } }
```

VPN 클라이언트에서 인터넷에 액세스하도록 허용

VPN을 통해 인터넷을 검색해야 하는 경우 추가 경로를 만들어야 합니다. 라우팅 테이블 탭을 선택한 다음 라우팅 생성를 선택합니다.

- 라우팅 대상 주소: 0.0.0.0/0
- Target VPC Subnet ID(대상 VPC 서브넷 ID): 인터넷 액세스 권한이 있는 연결된 서브넷 중 하나를 선택합니다.
- Create Route(라우팅 생성)를 선택합니다.

사용AWS CLI

다음 명령을 실행합니다.

```
aws ec2 create-client-vpn-route --client-vpn-endpoint-id cvpn-
endpoint-0123456789abcdefg --destination-cidr-block 0.0.0.0/0 --target-vpc-
subnet-id subnet-0123456789abdcdef
```

출력 예시:

```
{ "Status": { "Code": "creating" } }
```

VPN 클라이언트 구성

AWS Client VPN 대시보드에서 최근에 생성된 VPN 엔드포인트를 선택하고 클라이언트 구성 다운로드를 선택합니다. 구성 파일과 `easy-rsa/pki/issued/client1.domain.tld.crt` 및 `easy-rsa/pki/private/client1.domain.tld.key` 파일을 복사합니다. 구성 파일을 편집하고 다음 파라미터를 변경하거나 추가합니다.

- cert: `client1.domain.tld.crt` 파일을 가리키는 파라미터 인증서가 있는 새 줄을 추가합니다. 파일의 전체 경로를 사용합니다. 예시: `cert /home/user/.cert/client1.domain.tld.crt`
- cert: key: `client1.domain.tld.key` 파일을 가리키는 파라미터 키가 있는 새 줄을 추가합니다. 파일의 전체 경로를 사용합니다. 예시: `key /home/user/.cert/client1.domain.tld.key`

다음 명령을 사용하여 VPN 연결을 설정합니다. `sudo openvpn --config downloaded-client-config.ovpn`

액세스 취소

특정 클라이언트 키의 액세스를 무효화해야 하는 경우 CA에서 키를 취소해야 합니다. 그런 다음 취소 목록을 AWS Client VPN에 제출합니다.

easy-rsa로 키 취소:

- `cd easy-rsa`
- `./easyrsa3/easyrsa revoke client1.domain.tld`
- 계속하려면 “예”를 입력하고 중단하려면 다른 값을 입력합니다.

  ```
  Continue with revocation: `yes` ... * `./easyrsa3/easyrsa gen-crl
  ```
- 업데이트된 CRL이 생성되었습니다. CRL 파일: `/home/user/easy-rsa/pki/crl.pem`

취소 목록을 AWS Client VPN으로 가져오기:

- AWS Management 콘솔에서 서비스, VPC 순으로 선택합니다.
- Client VPN Endpoints(클라이언트 VPN 엔드포인트)를 선택합니다.

- Client VPN 엔드포인트를 선택한 다음 작업 -> Import Client Certificate CRL(클라이언트 인증서 CRL 가져오기)을 선택합니다.
- `crl.pem` 파일의 내용을 붙여넣습니다.

사용AWS CLI

다음 명령을 실행합니다.

```
aws ec2 import-client-vpn-client-certificate-revocation-list --certificate-revocation-list file://./easy-rsa/pki/crl.pem --client-vpn-endpoint-id cvpn-endpoint-0123456789abcdefg
```

출력 예시:

```
Example output: { "Return": true }
```

연결 엔드포인트 찾기

애플리케이션은 엔드포인트를 사용하여 클러스터에 연결합니다. 엔드포인트는 클러스터의 고유한 주소입니다. 모든 작업에 클러스터의 클러스터 엔드포인트를 사용합니다.

다음 섹션에서는 필요한 엔드포인트를 찾는 방법을 안내합니다.

MemoryDB 클러스터의 엔드포인트 찾기(AWS Management 콘솔)

MemoryDB 클러스터의 엔드포인트를 찾으려면

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.

   클러스터 목록이 포함된 클러스터 화면이 나타납니다. 연결하려는 클러스터를 선택합니다.
3. 클러스터의 엔드포인트를 찾으려면 클러스터 이름(라디오 버튼 아님)을 선택합니다.
4. 클러스터 엔드포인트는 클러스터 세부 정보 아래 표시됩니다. 이를 복사하려면 엔드포인트 왼쪽에 있는 복사 아이콘을 선택합니다.

MemoryDB 클러스터(CLI AWS)의 엔드포인트 찾기

`describe-clusters` 명령을 사용하여 클러스터의 엔드포인트를 찾을 수 있습니다. 명령은 클러스터의 엔드포인트를 반환합니다.

다음 작업은 클러스터 `mycluster`의 엔드포인트(이 예에서는 *##*로 표시됨)를 검색합니다.

다음과 같은 JSON 응답을 반환합니다.

```
aws memorydb describe-clusters \
  --cluster-name mycluster
```

Windows의 경우:

```
aws memorydb describe-clusters ^
   --cluster-name mycluster
```

```
{
    "Clusters": [
        {
            "Name": "my-cluster",
            "Status": "available",
            "NumberOfShards": 1,
            "ClusterEndpoint": {
                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
```

```
                "Port": 6379
            },
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.4",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "my-sg",
            "TLSEnabled": true,
            "ARN": "arn:aws:memorydb:us-east-1:zzzexamplearn:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
            "MaintenanceWindow": "wed:03:00-wed:04:00",
            "SnapshotWindow": "04:30-05:30",
            "ACLName": "my-acl",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```

자세한 내용은 describe-clusters 섹션을 참조하세요.

MemoryDB 클러스터의 엔드포인트 찾기(MemoryDB API)

MemoryDB API를 사용하여 클러스터의 엔드포인트를 찾을 수 있습니다.

MemoryDB 클러스터의 엔드포인트 찾기(MemoryDB API)

MemoryDB API를 사용하여 `DescribeClusters` 작업으로 클러스터의 엔드포인트를 찾을 수 있습니다. 작업은 클러스터의 엔드포인트를 반환합니다.

다음 작업은 클러스터 `mycluster`의 클러스터 엔드포인트를 검색합니다.

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DescribeClusters
    &ClusterName=mycluster
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210802T192317Z
    &Version=2021-01-01
    &X-Amz-Credential=<credential>
```

자세한 정보는 DescribeClusters를 참조하세요.

샤드 작업

샤드는 1~6개의 노드로 구성된 모음입니다. 하나의 클러스터당 최대 500개의 노드로 구성된 더 많은 수의 샤드와 더 적은 수의 복제본을 가진 클러스터를 생성할 수 있습니다. 이 클러스터 구성은 500개의 샤드 및 0개의 복제본부터 100개의 샤드 및 4개의 복제본까지 해당될 수 있으며, 이는 허용되는 최대 복제본 수입니다. 클러스터의 데이터는 클러스터의 샤드로 분할됩니다. 샤드에 둘 이상의 노드가 있는 경우 샤드는 한 노드가 읽기/쓰기 기본 노드가 되고 다른 노드가 읽기 전용 복제본 노드인 복제를 구현합니다.

AWS Management 콘솔을(를) 사용하여 MemoryDB 클러스터를 생성할 때 클러스터의 샤드 수와 샤드의 노드 수를 지정합니다. 자세한 내용은 MemoryDB 클러스터 생성 섹션을 참조하세요.

샤드의 각 노드는 컴퓨팅, 스토리지 및 메모리 사양이 동일합니다. MemoryDB API는 노드 수, 보안 설정 및 시스템 유지 관리 기간과 같은 클러스터 전체의 속성을 제어할 수 있도록 합니다.

자세한 내용은 MemoryDB의 오프라인 리샤딩 및 MemoryDB의 온라인 리샤딩(을)를 참조하세요.

샤드 이름 찾기

AWS Management 콘솔, AWS CLI 또는 MemoryDB API를 사용하여 샤드 이름을 검색할 수 있습니다.

AWS Management 콘솔 사용

다음 절차는 AWS Management 콘솔을 사용하여 MemoryDB 클러스터의 샤드 이름을 찾습니다.

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 좌측 탐색 창에서 클러스터(Clusters)를 선택합니다.
3. 이름에서 샤드 이름을 찾으려는 클러스터를 선택합니다.
4. 샤드 및 노드 탭의 이름 아래에서 샤드 목록을 확인합니다. 각 노드를 확장하여 해당 노드의 세부 정보를 볼 수도 있습니다.

AWS CLI 사용

MemoryDB 클러스터의 샤드(샤드) 이름을 찾으려면 다음 선택적 파라미터와 함께 AWS CLI 작업 `describe-clusters`를 사용하세요.

- **`--cluster-name`** - 사용되면 지정된 클러스터의 세부 정보 출력을 제한하는 선택적 파라미터입니다. 이 파라미터가 생략되면 최대 100개의 클러스터의 세부 정보가 반환됩니다.
- **`--show-shard-details`** - 샤드 이름을 포함한 샤드의 세부 정보를 반환합니다.

이 명령은 `my-cluster`의 세부 정보를 반환합니다.

Linux, macOS, Unix의 경우:

```
aws memorydb describe-clusters \
    --cluster-name my-cluster
    --show-shard-details
```

Windows의 경우:

```
aws memorydb describe-clusters ^
    --cluster-name my-cluster
    --show-shard-details
```

다음과 같은 JSON 응답이 반환됩니다.

줄바꿈은 가독성을 높이기 위해 추가되었습니다.

```
{
    "Clusters": [
        {
            "Name": "my-cluster",
            "Status": "available",
            "NumberOfShards": 1,
            "Shards": [
                {
                    "Name": "0001",
                    "Status": "available",
                    "Slots": "0-16383",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0001-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0001-002",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1b",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        }
                    ],
                    "NumberOfNodes": 2
                }
            ],
            "ClusterEndpoint": {
                "Address": "clustercfg.my-cluster.xxxxx.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
```

```
            },
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "my-sg",
            "TLSEnabled": true,
            "ARN": "arn:aws:memorydb:us-east-1:xxxxxexamplearn:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
            "MaintenanceWindow": "wed:03:00-wed:04:00",
            "SnapshotWindow": "04:30-05:30",
            "ACLName": "my-acl",
            "DataTiering": "false",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```

MemoryDB API 사용

MemoryDB 클러스터의 샤드 ID를 찾으려면 다음 선택적 파라미터와 함께 API 작업 `DescribeClusters`을(를) 사용합니다.

- **`ClusterName`** - 사용되면 지정된 클러스터의 세부 정보 출력을 제한하는 선택적 파라미터입니다. 이 파라미터가 생략되면 최대 100개의 클러스터의 세부 정보가 반환됩니다.
- **`ShowShardDetails`** - 샤드 이름을 포함한 샤드의 세부 정보를 반환합니다.

Example

이 명령은 `my-cluster`의 세부 정보를 반환합니다.

Linux, macOS, Unix의 경우:

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeClusters
   &ClusterName=sample-cluster
   &ShowShardDetails=true
   &Version=2021-01-01
   &SignatureVersion=4
```

```
    &SignatureMethod=HmacSHA256
    &Timestamp=20210802T192317Z
    &X-Amz-Credential=<credential>
```

MemoryDB 구현 관리

이 단원에서는 MemoryDB 구현의 다양한 구성 요소를 관리하는 방법에 관한 세부 정보를 얻을 수 있습니다.

주제

엔진 버전

이 섹션에서는 지원되는 Valkey 및 Redis OSS 엔진 버전을 설명합니다.

주제

MemoryDB 버전 7.3

2024년 12월 1일에 MemoryDB 7.3이 릴리스되었습니다. MemoryDB 버전 7.3은 다중 리전 클러스터를 지원하므로 지연 시간이 매우 짧고 가용성이 최대 99.999%인 다중 리전 애플리케이션을 구축할 수 있습니다. MemoryDB Multi-Region이 현재 지원되는 AWS 리전은 다음과 같습니다. 미국 동부(버지니아 북부 및 오하이오), 미국 서부(오리건, 캘리포니아 북부), 유럽(아일랜드, 프랑크푸르트, 런던), 아시아 태평양(도쿄, 시드니, 뭄바이, 서울, 싱가포르). 자세한 내용은 MemoryDB Multi-Region을 참조하세요.

MemoryDB 버전 7.2.6

2024년 10월 8일에 Valkey 7.2.6이 릴리스되었습니다. Valkey 7.2.6에는 Redis OSS 7.2.5 이전 버전과 유사한 호환성 차이가 있습니다. 다음은 Valkey와 Redis OSS 7.0 및 7.1 간의 주요 차이점입니다.

- ZRANK 및 ZREVRANK 명령에 대한 새로운 WITHSCORE 옵션
- CLIENT NO-TOUCH - 클라이언트가 키의 LRU/LFU에 영향을 주지 않고 명령을 실행할 수 있습니다.
- 새 명령 CLUSTER MYSHARDID는 복제를 기반으로 클러스터 모드에서 노드를 논리적으로 그룹화하기 위해 노드의 샤드 ID를 반환합니다.
- 다양한 데이터 유형에 대해 성능 및 메모리가 최적화됩니다.

다음은 Valkey 7.2와 Redis OSS 7.1(또는 7.0) 간의 잠재적 동작 변경 사항입니다.

- 동일한 채널을 구독하는 RESP3 클라이언트로 PUBLISH를 호출하면 순서가 변경되고 게시된 메시지 전에 회신이 전송됩니다.
- 스크립트에 대한 클라이언트 측 추적은 이제 EVAL/FCALL 호출자가 선언한 키 대신 스크립트가 읽는 키를 추적합니다.
- 정지 시간 샘플링은 명령 실행 중 및 스크립트에서 발생합니다.
- 차단된 명령이 차단 해제되면 ACL, OOM 등의 검사가 재평가됩니다.
- ACL 실패 오류 메시지 텍스트와 오류 코드가 통합됩니다.
- 키가 더 이상 존재하지 않을 때 해제되는 차단된 스트림 명령은 다른 오류 코드(-UNBLOCKED 대신 -NOGROUP 또는 -WRONGTYPE)를 전달합니다.
- 차단된 명령에 대해서는 명령이 실제로 실행될 때만 명령 통계가 업데이트됩니다.
- ACL 사용자의 내부 스토리지는 더 이상 중복 명령 및 범주 규칙을 제거하지 않습니다. 이렇게 하면 이러한 규칙이 ACL SAVE, ACL GETUSER 및 ACL LIST의 일부로 표시되는 방식이 변경될 수 있습니다.

- TLS 기반 복제를 위해 생성된 모든 클라이언트 연결은 가능한 경우 SNI를 사용합니다.
- XINFO STREAM: 이제 확인 시간 응답 필드에는 마지막으로 성공한 상호 작용 대신 마지막으로 시도한 상호 작용이 표시됩니다. 이제 새 활성 시간 응답 필드는 마지막으로 성공한 상호 작용을 나타냅니다.
- XREADGROUP 및 X[AUTO]CLAIM은 일부 읽기/클레임 수행 여부와 관계없이 소비자를 생성합니다.
- ACL 기본값으로 새로 생성된 사용자는 ACL LIST/GETUSER에서 sanitize-payload 플래그를 설정합니다.
- HELLO 명령은 성공하지 않는 한 클라이언트 상태에 영향을 미치지 않습니다.
- NAN 응답은 현재 inf 동작과 유사하게 단일 nan 유형으로 정규화됩니다.

Valkey에 대한 자세한 내용은 Valkey를 참조하세요.

Valkey 7.2 릴리스에 대한 자세한 내용은 GitHub의 Valkey에서 Redis OSS 7.2.4 릴리스 정보(Valkey 7.2에는 Redis OSS 버전 7.2.4까지의 모든 변경 사항이 포함됨) 및 Valkey 7.2 릴리스 정보를 참조하세요.

MemoryDB 버전 7.1(향상된 버전)

MemoryDB 버전 7.1에는 모든 리전의 벡터 검색 기능 지원과 중요한 버그 수정 및 성능 향상 기능이 추가되었습니다.

- 벡터 검색 기능: 기존 MemoryDB 기능에 벡터 검색을 사용할 수 있습니다. 벡터 검색을 사용하지 않는 애플리케이션은 이 기능의 영향을 받지 않습니다. 벡터 검색은 모든 리전에서 MemoryDB 버전 7.1 이상에 사용할 수 있습니다. 자세한 내용은 여기에 있는 설명서를 참조하세요.

> **Note**
>
> MemoryDB 버전 7.1은 Redis OSS v7.0과 호환됩니다. Redis OSS 7.0 릴리스에 대한 자세한 내용은 GitHub의 Redis OSS에서 Redis OSS 7.0 릴리스 정보를 참조하세요.

MemoryDB 버전 7.0(향상된 버전)

MemoryDB 7.0에는 여러 가지 개선 사항과 새로운 기능 지원이 추가되었습니다.

- 함수: MemoryDB 7에 함수 지원이 추가되었고, 연결할 때마다 클라이언트가 스크립트를 서버로 다시 전송할 필요 없이 개발자가 MemoryDB 클러스터에 저장된 애플리케이션 로직으로 LUA 스크립트를 실행할 수 있는 관리형 경험을 제공합니다.
- ACL 개선: MemoryDB 7에 차세대 버전의 액세스 제어 목록(ACL) 지원이 추가되었습니다. 이제 MemoryDB OSS Valkey 7 또는 Redis OSS 7에서는 클라이언트가 특정 키 또는 키스페이스에 다수의 권한 집합을 지정할 수 있습니다.
- 샤딩된 Pub/Sub: MemoryDB 7에서는 클러스터 모드 활성화(CME) 상태에서 MemoryDB를 실행할 때 샤딩된 방식으로 Pub/Sub 기능을 실행할 수 있도록 지원됩니다. 게시자는 Pub/Sub 기능으로 원하는 수의 채널 구독자에게 메시지를 발행할 수 있습니다. Amazon MemoryDB Valkey 7 및 Redis OSS 7을 사용하면, 채널이 MemoryDB 클러스터 내 샤드에 바인딩되므로 샤드 간에 채널 정보를 전파할 필요가 없어 확장성이 향상됩니다. 그 결과 확장성이 향상됩니다.
- 향상된 I/O 멀티플렉싱: MemoryDB Valkey 7 및 Redis OSS 버전 7에는 향상된 I/O 멀티플렉싱이 도입되어 MemoryDB 클러스터에 대한 동시 클라이언트 연결이 여럿이고 처리량이 많은 워크로드에 대해 처리량을 늘리고 지연 시간을 줄입니다. 예를 들어, r6g.4xlarge 노드로 구성된 클러스터를 사용하고 5200개의 동시 클라이언트를 실행하는 경우, MemoryDB 버전 6에 비해 처리량(초당 읽기 및 쓰기 작업)을 최대 46% 늘리고 P99 지연 시간을 최대 21% 줄일 수 있습니다.

Valkey에 대한 자세한 내용은 Valkey를 참조하세요.

Valkey 7.2 릴리스에 대한 자세한 내용은 GitHub의 Valkey에서 Redis OSS 7.2.4 릴리스 정보(Valkey 7.2에는 Redis OSS 버전 7.2.4까지의 모든 변경 사항이 포함됨) 및 Valkey 7.2 릴리스 정보를 참조하세요.

MemoryDB Redis OSS 버전 6.2(향상된 버전)

MemoryDB는 액세스 제어 목록(ACL)을 사용하여 사용자 인증, 자동 버전 업그레이드 지원, 클라이언트 측 캐싱 및 상당한 작동 성능 향상이 포함되어 있는 다음 버전의 Redis OSS 엔진을 새로 제공합니다.

Redis 엔진 버전 6.2.6에는 기본 JSON(JavaScript Object Notation) 형식에 대한 지원이 포함됩니다. 이 형식은 Redis OSS 클러스터 내에서 복잡한 데이터세트를 인코딩하는 간단한 스키마리스 방법입니다. JSON 지원으로 JSON을 통해 작동하는 애플리케이션의 성능 및 Redis OSS API를 활용할 수 있습니다. 자세한 내용은 JSON 시작하기 섹션을 참조하세요. 이 데이터 유형의 사용을 모니터링하기 위해 CloudWatch에 통합되는 JSON 관련 지표 `JsonBasedCmds`도 포함됩니다. 자세한 내용은 MemoryDB에 대한 지표 섹션을 참조하세요.

Redis OSS 6을 사용하면 MemoryDB는 여러 개의 패치 버전을 제공하는 대신 각 Redis OSS 마이너 릴리스의 단일 버전을 제공합니다. 이는 여러 마이너 버전 중에서 선택해야 하는 혼동과 모호성을 최소

화하도록 설계되었습니다. 또한 MemoryDB는 실행 중인 클러스터의 마이너 및 패치 버전을 자동으로 관리하여 개선된 성능과 향상된 보안을 보장합니다. 이러한 관리는 서비스 업데이트 캠페인을 사용하는 표준 고객 알림 채널을 통해 처리됩니다. 자세한 내용은 MemoryDB의 서비스 업데이트 섹션을 참조하세요.

생성 중에 엔진 버전을 지정하지 않으면 MemoryDB가 자동으로 선호하는 Redis OSS 버전을 선택합니다. 반면 `6.2`를 사용하여 엔진 버전을 지정하면 MemoryDB는 자동으로 사용 가능한 Redis OSS 6.2의 기본 패치 버전을 간접적으로 호출합니다.

예를 들어 캐시 클러스터를 생성할 때 `--engine-version` 파라미터를 `6.2`로 설정합니다. 클러스터는 생성 당시 사용 가능한 기본 패치 버전으로 시작됩니다. 전체 버전 값을 사용한 모든 요청이 거부되고 예외가 발생한 후 프로세스가 실패합니다.

`DescribeEngineVersions` API를 호출할 때 `EngineVersion` 파라미터 값이 6.2로 설정되고 실제 전체 엔진 버전은 `EnginePatchVersion` 필드에 반환됩니다.

Redis OSS 6.2 릴리스에 대한 자세한 내용은 GitHub의 Redis OSS에서 Redis 6.2 릴리스 정보를 참조하세요.

엔진 버전 업그레이드

MemoryDB는 기본적으로 서비스 업데이트를 통해 실행 중인 클러스터의 패치 버전을 자동으로 관리합니다. 클러스터의 `AutoMinorVersionUpgrade` 속성을 거짓으로 설정하는 경우, 자동 마이너 버전 업그레이드를 추가로 거부할 수 있습니다. 그러나 자동 패치 버전 업그레이드는 거부할 수 없습니다.

사용자는 클러스터를 실행하는 프로토콜 표준 소프트웨어를 자동 업그레이드가 시작되기 전에 MemoryDB에서 제공하는 새 버전으로 업그레이드할지 여부와 그 시기를 조정할 수 있습니다. 이 제어 수준을 사용하면 특정 버전과의 호환성을 유지하고, 프로덕션에 배포하기 전에 애플리케이션으로 새 버전을 테스트하고, 원하는 조건과 일정에 맞춰 버전 업그레이드를 수행할 수 있습니다.

Redis OSS 엔진이 있는 기존 MemoryDB에서 Valkey 엔진으로 업그레이드할 수도 있습니다.

다음과 같은 방법으로 클러스터의 엔진 버전 업그레이드를 시작할 수 있습니다.

- 업데이트하고 새 엔진 버전 지정. 자세한 내용은 MemoryDB 클러스터 수정 섹션을 참조하세요.
- 해당 엔진 버전에 대한 서비스 업데이트 적용. 자세한 내용은 MemoryDB의 서비스 업데이트 섹션을 참조하세요.

유의할 사항:

- 최신 엔진 버전으로 업그레이드할 수 있지만 이전 엔진 버전으로 다운그레이드할 수 없습니다. 이전 엔진 버전을 사용하려면, 기존 클러스터를 삭제하고 이전 엔진 버전을 통해 새로 생성해야 합니다.
- 대부분의 주요 개선 사항은 이전 버전으로 다시 포팅되지 않으므로 주기적으로 최신 메이저 버전으로 업그레이드하는 것이 좋습니다. MemoryDB가 새로운 AWS 리전으로 가용성을 확대함에 따라 MemoryDB는 새로운 리전에 대한 해당 시점의 최신 `MAJOR.MINOR` 버전 두 개를 지원합니다. 예를 들어 새로운 AWS 리전이 출시되고 최신 `MAJOR.MINOR` MemoryDB의 버전이 7.0 및 6.2인 경우, MemoryDB는 새로운 AWS 리전에서 버전 7.0 및 6.2를 지원합니다. 새로운 `MAJOR.MINOR` 버전의 MemoryDB가 릴리스됨에 따라 MemoryDB는 새로 릴리스되는 MemoryDB 버전에 대한 지원을 계속 추가할 것입니다. MemoryDB의 지역 선택에 대한 자세한 내용은 [지원되는 리전 및 엔드포인트](#)을 참조하세요.
- 엔진 버전 관리는 패치 발생 방법을 최대한 제어할 수 있도록 설계되었습니다. 그러나 MemoryDB는 시스템 또는 소프트웨어에 심각한 보안 취약성이 발견되는 등 발생할 가능성이 거의 없는 이벤트의 경우, 사용자를 대신하여 클러스터에 패치를 적용할 수 있는 권한을 보유합니다.
- MemoryDB는 여러 개의 패치 버전을 제공하는 대신 각 Valkey 또는 Redis OSS 마이너 릴리스의 단일 버전을 제공합니다. 이는 여러 버전 중에서 선택해야 하는 혼동과 모호성을 최소화하도록 설계되었습니다. 또한 MemoryDB는 실행 중인 클러스터의 마이너 및 패치 버전을 자동으로 관리하여 개선된 성능과 향상된 보안을 보장합니다. 이러한 관리는 서비스 업데이트 캠페인을 사용하는 표준 고객 알림 채널을 통해 처리됩니다. 자세한 내용은 [MemoryDB의 서비스 업데이트](#) 섹션을 참조하세요.
- 가동 중지 시간을 최소화하면서 클러스터 버전을 업그레이드할 수 있습니다. 전체 업그레이드 과정 중에도 클러스터를 읽을 수 있으며, 몇 초 정도 시간이 걸리는 장애 조치 작업 중인 경우를 제외하면 대부분 업그레이드 기간 중에 쓰기도 가능합니다.
- 들어오는 쓰기 트래픽이 적은 기간에 엔진 업그레이드를 수행하는 것이 좋습니다.

  샤드가 여러 개인 클러스터는 다음과 같이 처리되고 패치가 적용됩니다.

  - 언제든지 하나의 샤드당 오직 하나의 업그레이드 작업이 수행됩니다.
  - 각 샤드에서 기본 복제본이 처리되기 전에 모든 복제본이 처리됩니다. 하나의 샤드에 복제본이 적게 있는 경우에는 다른 샤드의 복제본의 처리가 완료되기 전에 해당 샤드의 기본 복제본이 처리됩니다.
  - 모든 샤드에서 프라이머리 노드가 연속하여 처리됩니다. 한번에 오직 하나의 프라이머리 노드가 업그레이드됩니다.

주제

- [엔진 버전 업그레이드 방법](#)
- [차단된 Redis OSS 엔진 업그레이드 문제 해결](#)

엔진 버전 업그레이드 방법

MemoryDB 콘솔, AWS CLI 또는 MemoryDB API를 통해 수정하고 최신 엔진 버전을 지정하여 클러스터에 대한 버전 업그레이드를 시작합니다. 자세한 내용은 다음 항목을 참조하세요.

- AWS Management 콘솔 사용
- AWS CLI 사용
- MemoryDB API 사용

차단된 Redis OSS 엔진 업그레이드 문제 해결

다음 표에 표시된 대로 대기 중인 스케일 업 작업이 있는 경우, Redis OSS 엔진 업그레이드 작업이 차단됩니다.

대기 중 작업	차단된 작업
스케일 업	즉시 엔진 업그레이드
엔진 업그레이드	즉시 스케일 업
스케일 업 및 엔진 업그레이드	즉시 스케일 업
	즉시 엔진 업그레이드

JSON 시작하기

MemoryDB는 Valkey 또는 Redis OSS 클러스터 내에서 복잡한 데이터세트를 인코딩하는 간단한 스키마리스 방법인 기본 JavaScript Object Notation(JSON) 형식을 지원합니다. 클러스터 내에서 JSON(JavaScript Object Notation) 형식을 사용하여 데이터를 저장하고 액세스하고, 직렬화 및 역직렬화를 위해 사용자 지정 코드를 관리할 필요 없이 해당 클러스터에 저장된 JSON 데이터를 업데이트할 수 있습니다.

JSON을 통해 작동하는 애플리케이션에 Valkey 또는 Redis OSS API를 활용하는 것 외에 전체 객체를 조작할 필요 없이 JSON 문서의 특정 부분을 효율적으로 검색하고 업데이트할 수 있으므로 성능이 향상되고 비용이 절감됩니다. Goessner 방식의 `JSONPath` 쿼리를 사용하여 JSON 문서 내용을 검색할 수도 있습니다.

지원되는 엔진 버전으로 클러스터를 생성하면 JSON 데이터 유형 및 관련 명령을 자동으로 사용할 수 있습니다. 이렇게 하면 RedisJSON 모듈의 버전 2와 API 및 RDB가 호환 가능하게 되므로 기존 JSON 기반 Valkey 또는 Redis OSS 애플리케이션을 MemoryDB로 쉽게 마이그레이션할 수 있습니다. 지원되는 명령에 대한 자세한 내용은 지원되는 명령 섹션을 참조하세요.

JSON 관련 지표 `JsonBasedCmds`는 CloudWatch에 통합되어 이 데이터 유형의 사용을 모니터링합니다. 자세한 정보는 MemoryDB 지표 섹션을 참조하세요.

> **Note**
>
> JSON을 사용하려면 Valkey 7.2 이상 또는 Redis OSS 엔진 버전 6.2.6 이상을 실행해야 합니다.

주제

- JSON 데이터 유형 개요
- 지원되는 명령

JSON 데이터 유형 개요

MemoryDB는 JSON 데이터 유형으로 작업하기 위해 다양한 Valkey 및 Redis OSS 명령을 지원합니다. 다음은 JSON 데이터 유형의 개요 및 지원되는 명령의 세부 목록입니다.

용어

Term	설명
JSON 문서	JSON 키의 값을 참조합니다.
JSON 값	전체 문서를 나타내는 루트를 포함하는 JSON 문서의 하위 집합을 참조합니다. 값은 컨테이너 또는 컨테이너 내의 항목일 수 있습니다.
JSON 요소	JSON 값과 같습니다.

지원되는 JSON 표준

JSON 형식은 RFC 7159 및 ECMA-404 JSON 데이터 교환 표준과 호환됩니다. JSON 형식 텍스트에서 UTF-8 유니코드가 지원됩니다.

루트 요소

루트 요소는 모든 JSON 데이터 유형일 수 있습니다. 이전 RFC 4627에서는 객체 또는 배열만 루트 값으로 허용되었습니다. RFC 7159로 업데이트한 이후 JSON 문서의 루트는 모든 JSON 데이터 유형이 될 수 있습니다.

문서 크기 제한

JSON 문서는 내부적으로 신속한 액세스 및 수정을 위해 최적화된 형식으로 저장됩니다. 이 형식은 일반적으로 동일한 문서의 동등하게 직렬화된 표현보다 어느 정도 더 많은 메모리를 소비하게 됩니다. 단일 JSON 문서에 의한 메모리 소비는 64MB로 제한됩니다. 이는 JSON 문자열이 아닌 메모리 내 데이터 구조의 크기입니다. `JSON.DEBUG MEMORY` 명령을 사용하여 JSON 문서가 소비하는 메모리 양을 확인할 수 있습니다.

JSON ACL

- JSON 데이터 유형은 Valkey 및 Redis OSS 액세스 제어 목록(ACL) 기능에 완전 통합됩니다. 기존의 데이터 유형별 범주(@string, @hash 등)와 유사합니다. JSON 명령 및 데이터에 대한 액세스 권한 관리를 단순화하기 위해 새 범주 @json이 추가되었습니다. 다른 기존 Valkey 또는 Redis OSS 명령은 @json 범주에 속하지 않습니다. 모든 JSON 명령은 모든 키스페이스 또는 명령 제한 및 권한을 적용합니다.
- @read, @write, @fast, @slow 및 @admin과 같은 새로운 JSON 명령을 포함하기 위해 업데이트된 다섯 가지 기존 ACL 범주가 있습니다. 다음의 표는 JSON 명령이 적절한 범주에 매핑되었음을 나타냅니다.

ACL

JSON 명령	@read	@write	@fast	@slow	@admin
JSON.ARRAPPEND		y	y		
JSON.ARRINDEX	y		y		

JSON 명령	@read	@write	@fast	@slow	@admin
JSON.ARRINSERT		y	y		
JSON.ARRLEN	y		y		
JSON.ARRPOP		y	y		
JSON.ARRTRIM		y	y		
JSON.CLEAR		y	y		
JSON.DEBUG	y			y	y
JSON.DEL		y	y		
JSON.FORGET		y	y		
JSON.GET	y		y		
JSON.MGET	y		y		
JSON.NUMINCRBY		y	y		
JSON.NUMMULTBY		y	y		
JSON.OBJKEYS	y		y		
JSON.OBJLEN	y		y		

JSON 명령	@read	@write	@fast	@slow	@admin
JSON.RESP	y		y		
JSON.SET		y		y	
JSON.STRAPPEND		y	y		
JSON.STRLEN	y		y		
JSON.STRLEN	y		y		
JSON.TOGGLE		y	y		
JSON.TYPE	y		y		
JSON.NUMINCRBY		y	y		

중첩 깊이 제한

JSON 객체 또는 배열에 자체로 또 다른 JSON 객체 또는 배열인 요소가 있는 경우, 해당 내부 객체 또는 배열은 외부 객체 또는 배열 내에 '중첩'된다고 합니다. 최대 중첩 깊이 제한은 128입니다. 중첩 깊이가 128보다 큰 문서를 만들려는 시도는 오류와 함께 거부됩니다.

명령 구문

대부분의 명령에는 첫 번째 인수로 Valkey 또는 Redis OSS 키 이름이 필요합니다. 일부 명령에는 path 인수도 있습니다. path 인수가 선택 사항이며 제공되지 않는 경우, 기본 루트로 설정됩니다.

표기법:

- 필수 인수는 각괄호로 묶습니다. 예: <키>
- 선택적 인수는 대괄호로 묶습니다. 예: [path]
- 추가 선택적 인수는 ... 로 표시됩니다 예: [json...]

경로 구문

Valkey 및 Redis OSS JSON은 다음과 같은 두 가지 종류의 경로 구문을 지원합니다.

- 향상된 구문 - 다음의 표에 표시된 대로 Goessner에서 설명하는 JSONPath 구문을 따릅니다. 명확하게 하기 위해 표의 설명을 재정렬하고 수정했습니다.
- 제한된 구문(Restricted syntax) - 쿼리 기능이 제한되었습니다.

> **Note**
> 일부 명령의 결과는 사용되는 경로 구문 유형에 민감합니다.

쿼리 경로가 '$'로 시작하는 경우 향상된 구문을 사용합니다. 그렇지 않으면 제한된 구문이 사용됩니다.

향상된 구문

기호/표현식	설명
$	루트 요소
. 또는 []	하위 연산자
..	재귀 하강
*	와일드카드 객체 또는 배열의 모든 요소.
[]	배열 아래 첨자 연산자 인덱스는 0부터 시작합니다.
[,]	조합 연산자
[start:end:step]	배열 조각 연산자
?()	현재 배열 또는 객체에 필터(스크립트) 표현식을 적용합니다.
()	필터 표현식

기호/표현식	설명
@	처리 중인 현재 노드를 참조하는 필터 표현식에 사용됩니다.
==	같음, 필터 표현식에 사용됩니다.
!=	같지 않음, 필터 표현식에 사용됩니다.
>	더 큼, 필터 표현식에 사용됩니다.
>=	더 크거나 같음, 필터 표현식에 사용됩니다.
<	더 작음, 필터 표현식에 사용됩니다.
<=	더 작거나 같음, 필터 표현식에 사용됩니다.
&&	논리적 AND, 여러 필터 표현식을 결합하는 데 사용됩니다.
||	논리적 OR, 여러 필터 표현식을 결합하는 데 사용됩니다.

예

다음의 예는 추가 필드를 추가하여 수정한 Goessner의 XML 데이터 예를 기반으로 구축되었습니다.

```
{ "store": {
    "book": [
      { "category": "reference",
        "author": "Nigel Rees",
        "title": "Sayings of the Century",
        "price": 8.95,
        "in-stock": true,
        "sold": true
      },
      { "category": "fiction",
        "author": "Evelyn Waugh",
        "title": "Sword of Honour",
        "price": 12.99,
        "in-stock": false,
```

```
        "sold": true
      },
      { "category": "fiction",
        "author": "Herman Melville",
        "title": "Moby Dick",
        "isbn": "0-553-21311-3",
        "price": 8.99,
        "in-stock": true,
        "sold": false
      },
      { "category": "fiction",
        "author": "J. R. R. Tolkien",
        "title": "The Lord of the Rings",
        "isbn": "0-395-19395-8",
        "price": 22.99,
        "in-stock": false,
        "sold": false
      }
    ],
    "bicycle": {
      "color": "red",
      "price": 19.95,
      "in-stock": true,
      "sold": false
    }
  }
}
```

경로	설명
$.store.book[*].author	상점에 있는 모든 책의 저자.
$..author	모든 저자.
$.store.*	상점의 모든 구성원.
$["store"].*	상점의 모든 구성원.
$.store..price	상점에 있는 모든 것의 가격.
$..*	JSON 구조의 모든 재귀 멤버.

경로	설명
$..book[*]	모든 책.
$..book[0]	첫 번째 책.
$..book[-1]	마지막 책.
$..book[0:2]	처음 두 권의 책.
$..book[0,1]	처음 두 권의 책.
$..book[0:4]	인덱스 0에서 3까지의 책(끝 인덱스는 포괄적이지 않음).
$..book[0:4:2]	인덱스 0, 2의 책.
$..book[?(@.isbn)]	ISBN 번호가 있는 모든 책.
$..book[?(@.price<10)]	모든 책이 10달러보다 저렴합니다.
'$..book[?(@.price < 10)]'	모든 책이 10달러보다 저렴합니다. (경로에 공백이 포함된 경우, 따옴표로 묶어야 합니다.)
'$..book[?(@["price"] < 10)]'	모든 책이 10달러보다 저렴합니다.
'$..book[?(@.["price"] < 10)]'	모든 책이 10달러보다 저렴합니다.
$..book[?(@.price>=10&&@.price<=100)]	가격대가 10달러에서 100달러인 모든 책, 포괄적.
'$..book[?(@.price>=10 && @.price<=100)]'	가격대가 10달러에서 100달러인 모든 책, 포괄적. (경로에 공백이 포함된 경우, 따옴표로 묶어야 합니다.)
$..book[?(@.sold==true||@.in-stock==false)]	모든 책이 매각 또는 품절되었습니다.
'$..book[?(@.sold == true || @.in-stock == false)]'	모든 책이 매각 또는 품절되었습니다. (경로에 공백이 포함된 경우, 따옴표로 묶어야 합니다.)
'$.store.book[?(@.["category"] == "fiction")]'	소설 범주의 모든 책.

경로	설명
'$.store.book[?(@.["category"] != "fiction")]'	비소설 범주의 모든 책.

추가 필터 표현식의 예:

```
127.0.0.1:6379> JSON.SET k1 . '{"books": [{"price":5,"sold":true,"in-
stock":true,"title":"foo"}, {"price":15,"sold":false,"title":"abc"}]}'
OK
127.0.0.1:6379> JSON.GET k1 $.books[?(@.price>1&&@.price<20&&@.in-stock)]
"[{\"price\":5,\"sold\":true,\"in-stock\":true,\"title\":\"foo\"}]"
127.0.0.1:6379> JSON.GET k1 '$.books[?(@.price>1 && @.price<20 && @.in-stock)]'
"[{\"price\":5,\"sold\":true,\"in-stock\":true,\"title\":\"foo\"}]"
127.0.0.1:6379> JSON.GET k1 '$.books[?((@.price>1 && @.price<20) && (@.sold==false))]'
"[{\"price\":15,\"sold\":false,\"title\":\"abc\"}]"
127.0.0.1:6379> JSON.GET k1 '$.books[?(@.title == "abc")]'
[{"price":15,"sold":false,"title":"abc"}]

127.0.0.1:6379> JSON.SET k2 . '[1,2,3,4,5]'
127.0.0.1:6379> JSON.GET k2 $.*.[?(@>2)]
"[3,4,5]"
127.0.0.1:6379> JSON.GET k2 '$.*.[?(@ > 2)]'
"[3,4,5]"

127.0.0.1:6379> JSON.SET k3 . '[true,false,true,false,null,1,2,3,4]'
OK
127.0.0.1:6379> JSON.GET k3 $.*.[?(@==true)]
"[true,true]"
127.0.0.1:6379> JSON.GET k3 '$.*.[?(@ == true)]'
"[true,true]"
127.0.0.1:6379> JSON.GET k3 $.*.[?(@>1)]
"[2,3,4]"
127.0.0.1:6379> JSON.GET k3 '$.*.[?(@ > 1)]'
"[2,3,4]"
```

제한된 구문(Restricted syntax

기호/표현식	설명
. 또는 []	하위 연산자.

기호/표현식	설명
[]	배열 아래 첨자 연산자. 인덱스는 0부터 시작합니다.

예시

경로	설명
.store.book[0].author	첫 번째 책의 저자.
.store.book[-1].author	마지막 책의 저자.
.address.city	도시 이름.
["store"]["book"][0]["title"]	첫 번째 책의 제목.
["store"]["book"][-1]["title"]	마지막 책의 제목.

> **Note**
> 이 문서에 인용된 모든 Goessner 콘텐츠는 크리에이티브 커먼즈 라이선스에 해당합니다.

일반적인 오류 접두사

각 오류 메시지에는 접두사가 있습니다. 다음은 일반적인 오류 접두사 목록입니다.

Prefix	설명
ERR	일반 오류.
LIMIT	크기 제한 초과 오류. 예: 문서 크기 제한 또는 중첩 깊이 제한을 초과했습니다.
NONEXISTENT	키 또는 경로가 존재하지 않습니다.
OUTOFBOUNDARIES	배열 인덱스가 범위를 벗어났습니다.

Prefix	설명
SYNTAXERR	구문 오류
WRONGTYPE	잘못된 값 유형.

JSON-관련 지표

다음과 같은 JSON 정보 지표가 제공됩니다.

정보	설명
json_total_memory_bytes	JSON 객체에 할당되는 총 메모리.
json_num_documents	Valkey 또는 Redis OSS 엔진의 총 문서 수

핵심 지표를 쿼리하려면 다음과 같은 명령을 실행합니다.

```
info json_core_metrics
```

메모리DB가 JSON과 상호작용하는 방식

다음은 MemoryDB가 JSON 데이터 유형과 상호 작용하는 방식을 보여줍니다.

연산자 우선순위

필터링에 대한 조건식을 평가하는 경우 &&s가 먼저 평가되고 그 다음 ||s가 평가되는데, 이는 대부분의 언어에서 일반적으로 사용됩니다. 괄호 안의 작업이 먼저 실행됩니다.

최대 경로 중첩 제한 동작

MemoryDB의 최대 경로 중첩 제한은 128입니다. 따라서 `$.a.b.c.d...`와 같은 값은 128수준까지만 도달할 수 있습니다.

숫자 값 처리

JSON에는 정수와 부동 소수점 숫자에 대해 별도의 데이터 유형이 없습니다. 모두 숫자라고 부릅니다.

JSON 번호를 수신하면 두 형식 중 하나로 저장됩니다. 숫자가 64비트 부호 있는 정수에 맞으면 해당 형식으로 변환되고 그렇지 않으면 문자열로 저장됩니다. 두 JSON 숫자(예: JSON.NUMINCRBY 및

JSON.NUMULTBY)에 대한 산술 연산은 정밀도를 최대한 유지하려고 합니다. 두 피연산자와 결과 값이 부호 있는 64비트 정수에 맞으면 정수 산술이 수행됩니다. 그렇지 않으면 입력 피연산자가 64비트 IEEE 배정밀도 부동 소수점 숫자로 변환되고 산술 연산이 수행되며 결과가 다시 문자열로 변환됩니다.

산술 명령어 `NUMINCRBY`과 `NUMMULTBY`:

- 두 숫자가 모두 정수이고 결과가 int64 범위를 벗어나는 경우, 자동으로 배정밀도 부동 소수점 숫자가 됩니다.
- 숫자 중 하나 이상이 부동 소수점인 경우, 결과는 배정밀도 부동 소수점 숫자가 됩니다.
- 결과가 두 배의 범위를 초과하면 명령은 `OVERFLOW` 오류를 반환합니다.

> **Note**
>
> Redis OSS 엔진 버전 6.2.6.R2 이전에는 입력으로 JSON 숫자를 받으면 64비트 부호 있는 정수 또는 64비트 IEEE 배정밀도 부동 소수점 중 하나의 내부 이진 표현으로 변환됩니다. 원래 문자열 및 모든해당 서식은 보관되지 않습니다. 따라서 JSON 응답의 일부로 숫자가 출력되면 해당 숫자는 내부 이진 표현에서 일반 서식 규칙을 사용하는 인쇄 가능한 문자열로 변환됩니다. 이러한 규칙은 수신된 문자열과 다른 문자열이 생성될 수 있습니다.
>
> - 두 숫자가 모두 정수이고 결과가 `int64`의 범위를 벗어나는 경우 자동으로 64비트 IEEE 배정밀도 부동 소수점 숫자가 됩니다.
> - 숫자 중 하나 이상이 부동 소수점인 경우 결과는 64비트 IEEE 배정밀도 부동 소수점 숫자입니다.
> - 결과가 64비트 IEEE 배정밀도 범위를 초과한 경우 명령은 `OVERFLOW` 오류를 반환합니다.

사용할 수 있는 명령의 자세한 목록은 [지원되는 명령](#) 섹션을 참조하세요.

엄격한 구문 평가

MemoryDB에서는 경로의 하위 집합에 유효한 경로가 포함되어 있더라도 잘못된 구문으로 JSON 경로를 허용하지 않습니다. 이는 고객에게 올바른 행동을 유지하는 것과 같습니다.

지원되는 명령

다음과 같은 JSON 명령이 지원됩니다.

주제

JSON.ARRAPPEND

하나 이상의 값을 경로의 배열 값에 추가합니다.

구문

```
JSON.ARRAPPEND <key> <path> <json> [json ...]
```

- 키(필수) - JSON 문서 유형의 키입니다.

- 경로(필수) - JSON 경로입니다.
- json(필수) - 배열에 추가할 JSON 값입니다.

반환

경로가 향상된 구문인 경우

- 각 경로에서 배열의 새 길이를 나타내는 정수 배열입니다.
- 값이 배열이 아닌 경우 해당 반환 값은 null입니다.
- SYNTAXERR입력 json 인수 중 하나가 유효한 JSON 문자열이 아닌 경우 오류가 발생합니다.
- 경로가 존재하지 않는 경우 NONEXISTENT 오류가 발생합니다.

경로가 제한된 구문인 경우

- 정수, 배열의 새 길이.
- 여러 배열 값을 선택한 경우 명령은 마지막으로 업데이트된 배열의 새 길이를 반환합니다.
- 경로의 값이 배열이 아닌 경우 WRONGTYPE 오류가 발생합니다.
- 입력 json 인수 중 하나가 유효한 JSON 문자열이 아닌 경우 SYNTAXERR 오류가 발생합니다.
- 경로가 존재하지 않는 경우 NONEXISTENT 오류가 발생합니다.

예

향상된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRAPPEND  k1 $[*] '"c"'
1) (integer) 1
2) (integer) 2
3) (integer) 3
127.0.0.1:6379> JSON.GET k1
"[[\"c\"],[\"a\",\"c\"],[\"a\",\"b\",\"c\"]]"
```

제한된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"]]'
```

```
OK
127.0.0.1:6379> JSON.ARRAPPEND  k1 [-1] '"c"'
(integer) 3
127.0.0.1:6379> JSON.GET k1
"[[],[\"a\"],[\"a\",\"b\",\"c\"]]"
```

JSON.ARRINDEX

경로의 배열에서 처음 나타나는 스칼라 JSON 값을 검색합니다.

- 범위를 벗어난 오류는 인덱스를 배열의 시작과 끝으로 반올림하여 처리됩니다.
- 시작 > 끝이면 -1(찾을 수 없음)을 반환합니다.

구문

```
JSON.ARRINDEX <key> <path> <json-scalar> [start [end]]
```

- 키(필수) - JSON 문서 유형의 키입니다.
- 경로(필수) - JSON 경로입니다.
- json-scalar (필수) - 검색할 스칼라 값입니다. JSON 스칼라는 객체나 배열이 아닌 값을 나타냅니다. 즉, 문자열, 숫자, 부울 및 null은 스칼라 값입니다.
- 시작(선택 사항) - 시작 인덱스는 포괄적입니다. 제공하지 않으면 기본적으로 0으로 설정됩니다.
- 끝(선택 사항) - 끝 인덱스는 배타적입니다. 제공되지 않은 경우, 기본적으로 0으로 설정됩니다. 즉, 마지막 요소가 포함된다는 의미입니다. 0 또는 -1은 마지막 요소가 포함되었음을 의미합니다.

반환

경로가 향상된 구문인 경우

- 정수 배열입니다. 각 값은 경로에 있는 배열에서 일치하는 요소의 인덱스입니다. 발견되지 않은 경우 값은 -1입니다.
- 값이 배열이 아닌 경우 해당 반환 값은 null입니다.

경로가 제한된 구문인 경우

- 정수이며, 일치하는 요소의 인덱스입니다. 또는 찾을 수 없는 경우 -1입니다.

- 경로의 값이 배열이 아닌 경우 WRONGTYPE 오류가 발생합니다.

예

향상된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"], ["a", "b", "c"]]'
OK
127.0.0.1:6379> JSON.ARRINDEX k1 $[*] '"b"'
1) (integer) -1
2) (integer) -1
3) (integer) 1
4) (integer) 1
```

제한된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . '{"children": ["John", "Jack", "Tom", "Bob", "Mike"]}'
OK
127.0.0.1:6379> JSON.ARRINDEX k1 .children '"Tom"'
(integer) 2
```

JSON.ARRINSERT

경로의 값 배열에서 인덱스 앞에 하나 이상의 값을 삽입합니다.

구문

```
JSON.ARRINSERT <key> <path> <index> <json> [json ...]
```

- 키(필수) - JSON 문서 유형의 키입니다.
- 경로(필수) - JSON 경로입니다.
- 인덱스(필수) - 삽입된 값의 앞에 있는 배열 인덱스입니다.
- json(필수) - 배열에 추가할 JSON 값입니다.

반환

경로가 향상된 구문인 경우

- 각 경로에서 배열의 새 길이를 나타내는 정수 배열입니다.
- 값이 빈 배열인 경우 해당 반환 값은 null입니다.
- 값이 배열이 아닌 경우 해당 반환 값은 null입니다.
- 인덱스 인수가 범위를 벗어난 경우 OUTOFBOUNDARIES 오류가 발생합니다.

경로가 제한된 구문인 경우

- 정수, 배열의 새 길이.
- 경로의 값이 배열이 아닌 경우 WRONGTYPE 오류가 발생합니다.
- 인덱스 인수가 범위를 벗어난 경우 OUTOFBOUNDARIES 오류가 발생합니다.

예

향상된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRINSERT k1 $[*] 0 '"c"'
1) (integer) 1
2) (integer) 2
3) (integer) 3
127.0.0.1:6379> JSON.GET k1
"[[\"c\"],[\"c\",\"a\"],[\"c\",\"a\",\"b\"]]"
```

제한된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRINSERT k1 . 0 '"c"'
(integer) 4
127.0.0.1:6379> JSON.GET k1
"[\"c\",[],[\"a\"],[\"a\",\"b\"]]"
```

JSON.ARRLEN

경로에서 배열 값의 길이를 얻습니다.

구문

```
JSON.ARRLEN <key> [path]
```

- 키(필수) - JSON 문서 유형의 키입니다.
- 경로(선택 사항) - JSON 경로입니다. 제공하지 않으면 기본적으로 root로 설정됩니다.

반환

경로가 향상된 구문인 경우

- 각 경로에서 배열 길이를 나타내는 정수 배열입니다.
- 값이 배열이 아닌 경우 해당 반환 값은 null입니다.
- 문서 키가 없으면 Null입니다.

경로가 제한된 구문인 경우

- 대량 문자열 배열 각 요소는 객체의 키 이름입니다.
- 정수, 배열 길이.
- 여러 객체를 선택한 경우 명령은 첫 번째 배열의 길이를 반환합니다.
- 경로의 값이 배열이 아닌 경우 `WRONGTYPE` 오류가 발생합니다.
- 경로가 존재하지 않는 경우 `WRONGTYPE` 오류가 발생합니다.
- 문서 키가 없으면 Null입니다.

예

향상된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . '[[], [\"a\"], [\"a\", \"b\"], [\"a\", \"b\", \"c\"]]'
(error) SYNTAXERR Failed to parse JSON string due to syntax error
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"], ["a", "b", "c"]]'
OK
127.0.0.1:6379> JSON.ARRLEN k1 $[*]
1) (integer) 0
2) (integer) 1
3) (integer) 2
4) (integer) 3
```

```
127.0.0.1:6379> JSON.SET k2 . '[[], "a", ["a", "b"], ["a", "b", "c"], 4]'
OK
127.0.0.1:6379> JSON.ARRLEN k2 $[*]
1) (integer) 0
2) (nil)
3) (integer) 2
4) (integer) 3
5) (nil)
```

제한된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"], ["a", "b", "c"]]'
OK
127.0.0.1:6379> JSON.ARRLEN k1 [*]
(integer) 0
127.0.0.1:6379> JSON.ARRLEN k1 $[3]
1) (integer) 3

127.0.0.1:6379> JSON.SET k2 . '[[], "a", ["a", "b"], ["a", "b", "c"], 4]'
OK
127.0.0.1:6379> JSON.ARRLEN k2 [*]
(integer) 0
127.0.0.1:6379> JSON.ARRLEN k2 $[1]
1) (nil)
127.0.0.1:6379> JSON.ARRLEN k2 $[2]
1) (integer) 2
```

JSON.ARRPOP

배열에서 인덱스의 요소를 제거하고 반환합니다. 빈 배열을 팝하면 null이 반환됩니다.

구문

```
JSON.ARRPOP <key> [path [index]]
```

- 키(필수) - JSON 문서 유형의 키입니다.
- 경로(선택 사항) - JSON 경로입니다. 제공하지 않으면 기본적으로 root로 설정됩니다.
- 인덱스(선택 사항) - 팝업을 시작할 배열의 위치입니다.

  - 제공되지 않으면 기본적으로 마지막 요소를 의미하는 -1로 설정됩니다.
  - 음수 값은 마지막 요소의 위치를 의미합니다.
  - 경계를 벗어난 인덱스는 각각의 배열 경계로 반올림됩니다.

반환

경로가 향상된 구문인 경우

- 각 경로에서 팝된 값을 나타내는 대량 문자열 배열입니다.
- 값이 빈 배열인 경우 해당 반환 값은 null입니다.
- 값이 배열이 아닌 경우 해당 반환 값은 null입니다.

경로가 제한된 구문인 경우

- 대량 문자열, 팝된 JSON 값을 나타냅니다.
- 배열인 비어 있는 경우 null입니다.
- 경로의 값이 배열이 아닌 경우 `WRONGTYPE` 오류가 발생합니다.

예

향상된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRPOP k1 $[*]
1) (nil)
2) "\"a\""
3) "\"b\""
127.0.0.1:6379> JSON.GET k1
"[[],[],[\"a\"]]"
```

제한된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRPOP k1
"[\"a\",\"b\"]"
```

```
127.0.0.1:6379> JSON.GET k1
"[[],[\"a\"]]"

127.0.0.1:6379> JSON.SET k2 . '[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRPOP k2 . 0
"[]"
127.0.0.1:6379> JSON.GET k2
"[[\"a\"],[\"a\",\"b\"]]"
```

JSON.ARRTRIM

경로의 배열을 자르므로 하위 배열[시작, 끝] 둘 다 포괄적이 됩니다.

- 배열이 비어 있는 경우 아무 것도 하지 않고 0을 반환합니다.
- 시작 < 0인 경우 0으로 처리합니다.
- 끝 >= 크기(배열의 크기)인 경우 크기-1로 처리합니다.
- 시작 >= 크기 또는 시작 > 끝인 경우 배열을 비우고 0을 반환합니다.

구문

```
JSON.ARRINSERT <key> <path> <start> <end>
```

- 키(필수) - JSON 문서 유형의 키입니다.
- 경로(필수) - JSON 경로입니다.
- 시작(필수) - 시작 인덱스는 포괄적입니다.
- 끝(필수) - 끝 인텍스, 포괄적입니다.

반환

경로가 향상된 구문인 경우

- 각 경로에서 배열의 새 길이를 나타내는 정수 배열입니다.
- 값이 빈 배열인 경우 해당 반환 값은 null입니다.
- 값이 배열이 아닌 경우 해당 반환 값은 null입니다.
- 인덱스 인수가 범위를 벗어난 경우 `OUTOFBOUNDARIES` 오류가 발생합니다.

경로가 제한된 구문인 경우

- 정수, 배열의 새 길이.
- 배열인 비어 있는 경우 null입니다.
- 경로의 값이 배열이 아닌 경우 `WRONGTYPE` 오류가 발생합니다.
- 인덱스 인수가 범위를 벗어난 경우 `OUTOFBOUNDARIES` 오류가 발생합니다.

예

향상된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"], ["a", "b", "c"]]'
OK
127.0.0.1:6379> JSON.ARRTRIM k1 $[*] 0 1
1) (integer) 0
2) (integer) 1
3) (integer) 2
4) (integer) 2
   127.0.0.1:6379> JSON.GET k1
   "[[],[\"a\"],[\"a\",\"b\"],[\"a\",\"b\"]]"
```

제한된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . '{"children": ["John", "Jack", "Tom", "Bob", "Mike"]}'
OK
127.0.0.1:6379> JSON.ARRTRIM k1 .children 0 1
(integer) 2
127.0.0.1:6379> JSON.GET k1 .children
"[\"John\",\"Jack\"]"
```

JSON.CLEAR

경로의 배열 또는 객체를 삭제합니다.

구문

```
JSON.CLEAR <key> [path]
```

- 키(필수) - JSON 문서 유형의 키입니다.
- 경로(선택 사항) - JSON 경로입니다. 제공하지 않으면 기본적으로 root로 설정됩니다.

반환

- 정수, 삭제된 컨테이너의 수입니다.
- 삭제된 0개의 컨테이너에 대해 빈 배열 또는 객체 계정을 삭제합니다.

> **Note**
>
> Redis OSS 버전 6.2.6.R2에서 삭제된 하나의 컨테이너에 대해 빈 배열 또는 객체 계정을 삭제합니다.

- 컨테이너가 아닌 값을 삭제하면 0이 반환됩니다.
- 경로 옆에 배열이나 개체 값이 없는 경우, 명령은 0을 반환합니다.

예시

```
127.0.0.1:6379> JSON.SET k1 . '[[], [0], [0,1], [0,1,2], 1, true, null, "d"]'
OK
127.0.0.1:6379>  JSON.CLEAR k1  $[*]
(integer) 6
127.0.0.1:6379> JSON.CLEAR k1  $[*]
(integer) 0
127.0.0.1:6379> JSON.SET k2 . '{"children": ["John", "Jack", "Tom", "Bob", "Mike"]}'
OK
127.0.0.1:6379> JSON.CLEAR k2 .children
(integer) 1
127.0.0.1:6379> JSON.GET k2 .children
"[]"
```

JSON.DEBUG

Report Information(보고서 정보) 지원되는 하위 명령은 다음과 같습니다.

- MEMORY <key> [path] - 메모리 사용량(단위: JSON 값 바이트)을 보고합니다. 경로는 제공되지 않으면 기본적으로 root로 설정됩니다.
- <key>DEPTH [path] - JSON 문서의 최대 경로 깊이를 보고합니다.

> **Note**
>
> 이 하위 명령은 Valkey 7.2 이상 또는 Redis OSS 엔진 버전 6.2.6.R2 이상에서만 사용할 수 있습니다.

- FIELDS <key> [path] - 지정된 문서 경로의 필드 수를 보고합니다. 경로는 제공되지 않으면 기본적으로 root로 설정됩니다. 컨테이너가 아닌 각 JSON 값은 하나의 필드로 계산됩니다. 객체와 배열은 포함하는 JSON 값 각각에 대해 하나의 필드를 재귀적으로 계산합니다. 루트 컨테이너를 제외한 각 컨테이너 값은 하나의 추가 필드로 계산됩니다.
- HELP - 명령의 도움말 메시지를 출력합니다.

구문

```
JSON.DEBUG <subcommand & arguments>
```

다음의 하위 명령에 따라 다릅니다.

MEMORY

- 경로가 향상된 구문인 경우
  - 각 경로에 있는 JSON 값의 메모리 크기(바이트)를 나타내는 정수로 구성된 배열을 반환합니다.
  - 키가 없으면 빈 배열을 반환합니다.
- 경로가 제한된 구문인 경우
  - 정수, 메모리 크기 및 JSON 값(단위: 바이트)을 반환합니다.
  - 키가 없으면 null을 반환합니다.

DEPTH

- JSON 문서의 최대 경로 깊이를 나타내는 정수를 반환합니다.
- 키가 없으면 null을 반환합니다.

FIELDS

- 경로가 향상된 구문인 경우
  - 각 경로에 있는 JSON 값의 필드 수를 나타내는 정수로 구성된 배열을 반반환합니다.

  - 키가 없으면 빈 배열을 반환합니다.
- 경로가 제한된 구문인 경우
  - JSON 값의 필드 개수인 정수를 반환합니다.
  - 키가 없으면 null을 반환합니다.

HELP - 도움말 메시지 배열을 반환합니다.

예

향상된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . '[1, 2.3, "foo", true, null, {}, [], {"a":1, "b":2},
 [1,2,3]]'
OK
127.0.0.1:6379> JSON.DEBUG MEMORY k1 $[*]
1) (integer) 16
2) (integer) 16
3) (integer) 19
4) (integer) 16
5) (integer) 16
6) (integer) 16
7) (integer) 16
8) (integer) 50
9) (integer) 64
127.0.0.1:6379> JSON.DEBUG FIELDS k1 $[*]
1) (integer) 1
2) (integer) 1
3) (integer) 1
4) (integer) 1
5) (integer) 1
6) (integer) 0
7) (integer) 0
8) (integer) 2
9) (integer) 3
```

제한된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 .
 '{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
```

```
 York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"},{"type":"office","number":"646
 555-4567"}],"children":[],"spouse":null}'
OK
127.0.0.1:6379> JSON.DEBUG MEMORY k1
(integer) 632
127.0.0.1:6379> JSON.DEBUG MEMORY k1 .phoneNumbers
(integer) 166

127.0.0.1:6379> JSON.DEBUG FIELDS k1
(integer) 19
127.0.0.1:6379> JSON.DEBUG FIELDS k1 .address
(integer) 4

127.0.0.1:6379> JSON.DEBUG HELP
1) JSON.DEBUG MEMORY <key> [path] - report memory size (bytes) of the JSON element.
 Path defaults to root if not provided.
2) JSON.DEBUG FIELDS <key> [path] - report number of fields in the JSON element. Path
 defaults to root if not provided.
3) JSON.DEBUG HELP - print help message.
```

JSON.DEL

문서 키의 경로에 있는 JSON 값을 삭제합니다. 경로가 루트이면 Valkey 또는 Redis OSS에서 키를 삭제하는 것과 같습니다.

구문

```
JSON.DEL <key> [path]
```

- 키(필수) - JSON 문서 유형의 키입니다.
- 경로(선택 사항) - JSON 경로입니다. 제공하지 않으면 기본적으로 root로 설정됩니다.

반환

- 삭제된 요소 수입니다.
- 키가 없으면 0입니다.
- JSON 경로가 잘못되었거나 존재하지 않는 경우 0입니다.

예

향상된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1,
 "b":2, "c":3}, "e": [1,2,3,4,5]}'
OK
127.0.0.1:6379> JSON.DEL k1 $.d.*
(integer) 3
127.0.0.1:6379> JSOn.GET k1
"{\"a\":{},\"b\":{\"a\":1},\"c\":{\"a\":1,\"b\":2},\"d\":{},\"e\":[1,2,3,4,5]}"
127.0.0.1:6379> JSON.DEL k1 $.e[*]
(integer) 5
127.0.0.1:6379> JSOn.GET k1
"{\"a\":{},\"b\":{\"a\":1},\"c\":{\"a\":1,\"b\":2},\"d\":{},\"e\":[]}"
```

제한된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1,
 "b":2, "c":3}, "e": [1,2,3,4,5]}'
OK
127.0.0.1:6379> JSON.DEL k1 .d.*
(integer) 3
127.0.0.1:6379> JSON.GET k1
"{\"a\":{},\"b\":{\"a\":1},\"c\":{\"a\":1,\"b\":2},\"d\":{},\"e\":[1,2,3,4,5]}"
127.0.0.1:6379> JSON.DEL k1 .e[*]
(integer) 5
127.0.0.1:6379> JSON.GET k1
"{\"a\":{},\"b\":{\"a\":1},\"c\":{\"a\":1,\"b\":2},\"d\":{},\"e\":[]}"
```

JSON.FORGET

JSON.DEL의 별칭.

JSON.GET

하나 또는 여러 경로에서 직렬화된 JSON을 반환합니다.

구문

```
JSON.GET <key>
```

```
[INDENT indentation-string]
[NEWLINE newline-string]
[SPACE space-string]
[NOESCAPE]
[path ...]
```

- 키(필수) - JSON 문서 유형의 키입니다.
- INDENT/NEWLINE/SPACE(선택 사항) - 반환된 JSON 문자열(즉, "예쁜 인쇄")의 형식을 제어합니다. 각 문자열의 기본값은 빈 문자열이며, 모든 조합으로 재정의될 수 있습니다. 임의의 순서로 지정할 수 있습니다.
- NOESCAPE - 선택 사항, 레거시 호환성을 위해 사용할 수 있으며 다른 효과는 없습니다.
- 경로(선택 사항) - 0개 이상의 JSON 경로, 아무 것도 제공되지 않는 경우 기본적으로 루트로 설정됩니다. 경로 인수는 끝에 배치해야 합니다.

반환

향상된 경로 구문.

경로가 하나 지정된 경우,

- 값의 배열의 직렬화된 문자열을 반환합니다.
- 값을 선택하지 않은 경우 명령은 빈 배열을 반환합니다.

여러 경로가 지정된 경우,

- 각 경로가 키인 문자열화된 JSON 객체를 반환합니다.
- 향상된 경로 구문과 제한된 경로 구문이 혼합된 경우 결과는 향상된 구문을 따릅니다.
- 경로가 없는 경우 해당 값은 빈 배열입니다.

예

향상된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 .
 '{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
```

```
[{"type":"home","number":"212 555-1234"},{"type":"office","number":"646
 555-4567"}],"children":[],"spouse":null}'
OK
127.0.0.1:6379> JSON.GET k1 $.address.*
"[\"21 2nd Street\",\"New York\",\"NY\",\"10021-3100\"]"
127.0.0.1:6379> JSON.GET k1 indent "\t" space " " NEWLINE "\n" $.address.*
"[\n\t\"21 2nd Street\",\n\t\"New York\",\n\t\"NY\",\n\t\"10021-3100\"\n]"
127.0.0.1:6379> JSON.GET k1 $.firstName $.lastName $.age
"{\"$.firstName\":[\"John\"],\"$.lastName\":[\"Smith\"],\"$.age\":[27]}"
127.0.0.1:6379> JSON.SET k2 . '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}}'
OK
127.0.0.1:6379> json.get k2 $..*
"[{},{\"a\":1},{\"a\":1,\"b\":2},1,1,2]"
```

제한된 경로 구문.

```
 127.0.0.1:6379> JSON.SET k1 .
 '{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"},{"type":"office","number":"646
 555-4567"}],"children":[],"spouse":null}'
OK
127.0.0.1:6379> JSON.GET k1 .address
"{\"street\":\"21 2nd Street\",\"city\":\"New York\",\"state\":\"NY\",\"zipcode\":
\"10021-3100\"}"
127.0.0.1:6379> JSON.GET k1 indent "\t" space " " NEWLINE "\n" .address
"{\n\t\"street\": \"21 2nd Street\",\n\t\"city\": \"New York\",\n\t\"state\": \"NY\",\n
\t\"zipcode\": \"10021-3100\"\n}"
127.0.0.1:6379> JSON.GET k1 .firstName .lastName .age
"{\".firstName\":\"John\",\".lastName\":\"Smith\",\".age\":27}"
```

JSON.MGET

여러 문서 키의 경로에서 직렬화된 JSON을 얻습니다. 없는 키 또는 JSON 경로에 대해 null을 반환합니다.

구문

```
JSON.MGET <key> [key ...] <path>
```

- 키(필수) - 문서 유형의 하나 이상의 키입니다.
- 경로(필수) - JSON 경로입니다.

반환

- 대량 문자열 배열 배열의 크기는 명령의 키 수와 같습니다. 키가 없는 경우, 경로가 문서에 없는 경우, 또는 경로가 잘못된 경우(구문 오류), 배열의 각 요소는 (a) 경로에 있는 직렬화된 JSON 또는 (b) null로 채워집니다.
- 지정된 키 중 하나라도 있고 JSON 키가 아닌 경우 명령은 `WRONGTYPE` 오류를 반환합니다.

예

향상된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . '{"address":{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021"}}'
OK
127.0.0.1:6379> JSON.SET k2 . '{"address":{"street":"5 main
 Street","city":"Boston","state":"MA","zipcode":"02101"}}'
OK
127.0.0.1:6379> JSON.SET k3 . '{"address":{"street":"100 Park
 Ave","city":"Seattle","state":"WA","zipcode":"98102"}}'
OK
127.0.0.1:6379> JSON.MGET k1 k2 k3 $.address.city
1) "[\"New York\"]"
2) "[\"Boston\"]"
3) "[\"Seattle\"]"
```

제한된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . '{"address":{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021"}}'
OK
127.0.0.1:6379> JSON.SET k2 . '{"address":{"street":"5 main
 Street","city":"Boston","state":"MA","zipcode":"02101"}}'
OK
127.0.0.1:6379> JSON.SET k3 . '{"address":{"street":"100 Park
 Ave","city":"Seattle","state":"WA","zipcode":"98102"}}'
OK
```

```
127.0.0.1:6379> JSON.MGET k1 k2 k3 .address.city
1) "\"New York\""
2) "\"Seattle\""
3) "\"Seattle\""
```

JSON.NUMINCRBY

경로의 숫자 값을 지정된 수만큼 증가합니다.

구문

```
JSON.NUMINCRBY <key> <path> <number>
```

- 키(필수) - JSON 문서 유형의 키입니다.
- 경로(필수) - JSON 경로입니다.
- 숫자(필수) - 숫자입니다.

반환

경로가 향상된 구문인 경우

- 각 경로에서 결과 값을 나타내는 대량 문자열 배열입니다.
- 값이 숫자가 아닌 경우 해당 반환 값은 null입니다.
- 숫자를 구문 분석할 수 없는 경우 `WRONGTYPE` 오류가 발생합니다.
- 결과가 64비트 IEEE 배정밀도 범위를 벗어나는 경우 `OVERFLOW` 오류가 발생합니다.
- 문서 키가 없는 경우 `NONEXISTENT`입니다.

경로가 제한된 구문인 경우

- 결과 값을 나타내는 대량 문자열입니다.
- 여러 값을 선택한 경우 명령은 마지막으로 업데이트된 값의 결과를 반환합니다.
- 경로의 값이 숫자가 아닌 경우 `WRONGTYPE` 오류가 발생합니다.
- 숫자를 구문 분석할 수 없는 경우 `WRONGTYPE` 오류가 발생합니다.
- 결과가 64비트 IEEE 배정밀도 범위를 벗어나는 경우 `OVERFLOW` 오류가 발생합니다.

- 문서 키가 없는 경우 NONEXISTENT입니다.

예

향상된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k1 $.d[*] 10
"[11,12,13]"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[1],\"c\":[1,2],\"d\":[11,12,13]}"

127.0.0.1:6379> JSON.SET k1 $ '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k1 $.a[*] 1
"[]"
127.0.0.1:6379> JSON.NUMINCRBY k1 $.b[*] 1
"[2]"
127.0.0.1:6379> JSON.NUMINCRBY k1 $.c[*] 1
"[2,3]"
127.0.0.1:6379> JSON.NUMINCRBY k1 $.d[*] 1
"[2,3,4]"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[2,3],\"d\":[2,3,4]}"

127.0.0.1:6379> JSON.SET k2 $ '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1,
 "b":2, "c":3}}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k2 $.a.* 1
"[]"
127.0.0.1:6379> JSON.NUMINCRBY k2 $.b.* 1
"[2]"
127.0.0.1:6379> JSON.NUMINCRBY k2 $.c.* 1
"[2,3]"
127.0.0.1:6379> JSON.NUMINCRBY k2 $.d.* 1
"[2,3,4]"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":2,\"b\":3},\"d\":{\"a\":2,\"b\":3,\"c\":4}}"

127.0.0.1:6379> JSON.SET k3 $ '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"b"}, "d":{"a":1, "b":"b", "c":3}}'
OK
```

```
127.0.0.1:6379> JSON.NUMINCRBY k3 $.a.* 1
"[null]"
127.0.0.1:6379> JSON.NUMINCRBY k3 $.b.* 1
"[null,2]"
127.0.0.1:6379> JSON.NUMINCRBY k3 $.c.* 1
"[null,null]"
127.0.0.1:6379> JSON.NUMINCRBY k3 $.d.* 1
"[2,null,4]"
127.0.0.1:6379> JSON.GET k3
"{\"a\":{\"a\":\"a\"},\"b\":{\"a\":\"a\",\"b\":2},\"c\":{\"a\":\"a\",\"b\":\"b\"},\"d
\":{\"a\":2,\"b\":\"b\",\"c\":4}}"
```

제한된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k1 .d[1] 10
"12"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[1],\"c\":[1,2],\"d\":[1,12,3]}"

127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k1 .a[*] 1
(error) NONEXISTENT JSON path does not exist
127.0.0.1:6379> JSON.NUMINCRBY k1 .b[*] 1
"2"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[1,2],\"d\":[1,2,3]}"
127.0.0.1:6379> JSON.NUMINCRBY k1 .c[*] 1
"3"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[2,3],\"d\":[1,2,3]}"
127.0.0.1:6379> JSON.NUMINCRBY k1 .d[*] 1
"4"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[2,3],\"d\":[2,3,4]}"

127.0.0.1:6379> JSON.SET k2 . '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1,
 "b":2, "c":3}}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k2 .a.* 1
```

```
(error) NONEXISTENT JSON path does not exist
127.0.0.1:6379> JSON.NUMINCRBY k2 .b.* 1
"2"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":1,\"b\":2},\"d\":{\"a\":1,\"b\":2,\"c\":3}}"
127.0.0.1:6379> JSON.NUMINCRBY k2 .c.* 1
"3"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":2,\"b\":3},\"d\":{\"a\":1,\"b\":2,\"c\":3}}"
127.0.0.1:6379> JSON.NUMINCRBY k2 .d.* 1
"4"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":2,\"b\":3},\"d\":{\"a\":2,\"b\":3,\"c\":4}}"

127.0.0.1:6379> JSON.SET k3 . '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"b"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k3 .a.* 1
(error) WRONGTYPE JSON element is not a number
127.0.0.1:6379> JSON.NUMINCRBY k3 .b.* 1
"2"
127.0.0.1:6379> JSON.NUMINCRBY k3 .c.* 1
(error) WRONGTYPE JSON element is not a number
127.0.0.1:6379> JSON.NUMINCRBY k3 .d.* 1
"4"
```

JSON.NUMMULTBY

경로의 숫자 값을 지정된 수만큼 곱합니다.

구문

```
JSON.NUMMULTBY <key> <path> <number>
```

- 키(필수) - JSON 문서 유형의 키입니다.
- 경로(필수) - JSON 경로입니다.
- 숫자(필수) - 숫자입니다.

반환

경로가 향상된 구문인 경우

- 각 경로에서 결과 값을 나타내는 대량 문자열 배열입니다.
- 값이 숫자가 아닌 경우 해당 반환 값은 null입니다.
- 숫자를 구문 분석할 수 없는 경우 `WRONGTYPE` 오류가 발생합니다.
- 결과가 64비트 IEEE 배정밀도 범위를 벗어나는 경우 `OVERFLOW` 오류가 발생합니다.
- 문서 키가 없는 경우 `NONEXISTENT`입니다.

경로가 제한된 구문인 경우

- 결과 값을 나타내는 대량 문자열입니다.
- 여러 값을 선택한 경우 명령은 마지막으로 업데이트된 값의 결과를 반환합니다.
- 경로의 값이 숫자가 아닌 경우 `WRONGTYPE` 오류가 발생합니다.
- 숫자를 구문 분석할 수 없는 경우 `WRONGTYPE` 오류가 발생합니다.
- 결과가 64비트 IEEE 배정밀도 범위를 벗어나는 경우 `OVERFLOW` 오류가 발생합니다.
- 문서 키가 없는 경우 `NONEXISTENT`입니다.

예

향상된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k1 $.d[*] 2
"[2,4,6]"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[1],\"c\":[1,2],\"d\":[2,4,6]}"

127.0.0.1:6379> JSON.SET k1 $ '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k1 $.a[*] 2
"[]"
127.0.0.1:6379> JSON.NUMMULTBY k1 $.b[*] 2
"[2]"
127.0.0.1:6379> JSON.NUMMULTBY k1 $.c[*] 2
"[2,4]"
127.0.0.1:6379> JSON.NUMMULTBY k1 $.d[*] 2
"[2,4,6]"
```

```
127.0.0.1:6379> JSON.SET k2 $ '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1,
 "b":2, "c":3}}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k2 $.a.* 2
"[]"
127.0.0.1:6379> JSON.NUMMULTBY k2 $.b.* 2
"[2]"
127.0.0.1:6379> JSON.NUMMULTBY k2 $.c.* 2
"[2,4]"
127.0.0.1:6379> JSON.NUMMULTBY k2 $.d.* 2
"[2,4,6]"

127.0.0.1:6379> JSON.SET k3 $ '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"b"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k3 $.a.* 2
"[null]"
127.0.0.1:6379> JSON.NUMMULTBY k3 $.b.* 2
"[null,2]"
127.0.0.1:6379> JSON.NUMMULTBY k3 $.c.* 2
"[null,null]"
127.0.0.1:6379> JSON.NUMMULTBY k3 $.d.* 2
"[2,null,6]"
```

제한된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k1 .d[1] 2
"4"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[1],\"c\":[1,2],\"d\":[1,4,3]}"

127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k1 .a[*] 2
(error) NONEXISTENT JSON path does not exist
127.0.0.1:6379> JSON.NUMMULTBY k1 .b[*] 2
"2"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[1,2],\"d\":[1,2,3]}"
```

```
127.0.0.1:6379> JSON.NUMMULTBY k1 .c[*] 2
"4"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[2,4],\"d\":[1,2,3]}"
127.0.0.1:6379> JSON.NUMMULTBY k1 .d[*] 2
"6"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[2,4],\"d\":[2,4,6]}"

127.0.0.1:6379> JSON.SET k2 . '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1,
 "b":2, "c":3}}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k2 .a.* 2
(error) NONEXISTENT JSON path does not exist
127.0.0.1:6379> JSON.NUMMULTBY k2 .b.* 2
"2"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":1,\"b\":2},\"d\":{\"a\":1,\"b\":2,\"c\":3}}"
127.0.0.1:6379> JSON.NUMMULTBY k2 .c.* 2
"4"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":2,\"b\":4},\"d\":{\"a\":1,\"b\":2,\"c\":3}}"
127.0.0.1:6379> JSON.NUMMULTBY k2 .d.* 2
"6"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":2,\"b\":4},\"d\":{\"a\":2,\"b\":4,\"c\":6}}"

127.0.0.1:6379> JSON.SET k3 . '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"b"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k3 .a.* 2
(error) WRONGTYPE JSON element is not a number
127.0.0.1:6379> JSON.NUMMULTBY k3 .b.* 2
"2"
127.0.0.1:6379> JSON.GET k3
"{\"a\":{\"a\":\"a\"},\"b\":{\"a\":\"a\",\"b\":2},\"c\":{\"a\":\"a\",\"b\":\"b\"},\"d
\":{\"a\":1,\"b\":\"b\",\"c\":3}}"
127.0.0.1:6379> JSON.NUMMULTBY k3 .c.* 2
(error) WRONGTYPE JSON element is not a number
127.0.0.1:6379> JSON.NUMMULTBY k3 .d.* 2
"6"
127.0.0.1:6379> JSON.GET k3
"{\"a\":{\"a\":\"a\"},\"b\":{\"a\":\"a\",\"b\":2},\"c\":{\"a\":\"a\",\"b\":\"b\"},\"d
\":{\"a\":2,\"b\":\"b\",\"c\":6}}"
```

JSON.OBJLEN

경로에 있는 객체 값의 키 수를 얻습니다.

구문

```
JSON.OBJLEN <key> [path]
```

- 키(필수) - JSON 문서 유형의 키입니다.
- 경로(선택 사항) - JSON 경로입니다. 제공하지 않으면 기본적으로 root로 설정됩니다.

반환

경로가 향상된 구문인 경우

- 각 경로에서 객체 길이를 나타내는 정수 배열입니다.
- 값이 객체가 아닌 경우 해당 반환 값은 null입니다.
- 문서 키가 없으면 null입니다.

경로가 제한된 구문인 경우

- 정수, 객체의 키 수입니다.
- 여러 객체를 선택한 경우 명령은 첫 번째 객체의 길이를 반환합니다.
- 경로의 값이 객체가 아닌 경우 `WRONGTYPE` 오류가 발생합니다.
- 경로가 존재하지 않는 경우 `WRONGTYPE` 오류가 발생합니다.
- 문서 키가 없으면 Null입니다.

예

향상된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{}, "b":{"a":"a"}, "c":{"a":"a", "b":"bb"}, "d":
{"a":1, "b":"b", "c":{"a":3,"b":4}}, "e":1}'
OK
127.0.0.1:6379> JSON.OBJLEN k1 $.a
```

```
1) (integer) 0
127.0.0.1:6379> JSON.OBJLEN k1 $.a.*
(empty array)
127.0.0.1:6379> JSON.OBJLEN k1 $.b
1) (integer) 1
127.0.0.1:6379> JSON.OBJLEN k1 $.b.*
1) (nil)
127.0.0.1:6379> JSON.OBJLEN k1 $.c
1) (integer) 2
127.0.0.1:6379> JSON.OBJLEN k1 $.c.*
1) (nil)
2) (nil)
127.0.0.1:6379> JSON.OBJLEN k1 $.d
1) (integer) 3
127.0.0.1:6379> JSON.OBJLEN k1 $.d.*
1) (nil)
2) (nil)
3) (integer) 2
127.0.0.1:6379> JSON.OBJLEN k1 $.*
1) (integer) 0
2) (integer) 1
3) (integer) 2
4) (integer) 3
5) (nil)
```

제한된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . '{"a":{}, "b":{"a":"a"}, "c":{"a":"a", "b":"bb"}, "d":
{"a":1, "b":"b", "c":{"a":3,"b":4}}, "e":1}'
OK
127.0.0.1:6379> JSON.OBJLEN k1 .a
(integer) 0
127.0.0.1:6379> JSON.OBJLEN k1 .a.*
(error) NONEXISTENT JSON path does not exist
127.0.0.1:6379> JSON.OBJLEN k1 .b
(integer) 1
127.0.0.1:6379> JSON.OBJLEN k1 .b.*
(error) WRONGTYPE JSON element is not an object
127.0.0.1:6379> JSON.OBJLEN k1 .c
(integer) 2
127.0.0.1:6379> JSON.OBJLEN k1 .c.*
(error) WRONGTYPE JSON element is not an object
```

```
127.0.0.1:6379> JSON.OBJLEN k1 .d
(integer) 3
127.0.0.1:6379> JSON.OBJLEN k1 .d.*
(integer) 2
127.0.0.1:6379> JSON.OBJLEN k1 .*
(integer) 0
```

JSON.OBJKEYS

경로의 객체 값에 있는 키 이름을 얻습니다.

구문

```
JSON.OBJKEYS <key> [path]
```

- 키(필수) - JSON 문서 유형의 키입니다.
- 경로(선택 사항) - JSON 경로입니다. 제공하지 않으면 기본적으로 root로 설정됩니다.

반환

경로가 향상된 구문인 경우

- 대량 문자열 배열 각 요소는 일치하는 객체의 키 배열입니다.
- 값이 객체가 아닌 경우 해당 반환 값은 빈 값입니다.
- 문서 키가 없으면 null입니다.

경로가 제한된 구문인 경우

- 대량 문자열 배열 각 요소는 객체의 키 이름입니다.
- 여러 객체를 선택한 경우 명령은 첫 번째 객체의 키를 반환합니다.
- 경로의 값이 객체가 아닌 경우 `WRONGTYPE` 오류가 발생합니다.
- 경로가 존재하지 않는 경우 `WRONGTYPE` 오류가 발생합니다.
- 문서 키가 없으면 Null입니다.

예

향상된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{}, "b":{"a":"a"}, "c":{"a":"a", "b":"bb"}, "d":
{"a":1, "b":"b", "c":{"a":3,"b":4}}, "e":1}'
OK
127.0.0.1:6379> JSON.OBJKEYS k1 $.*
1) (empty array)
2) 1) "a"
3) 1) "a"
   2) "b"
4) 1) "a"
   2) "b"
   3) "c"
5) (empty array)
127.0.0.1:6379> JSON.OBJKEYS k1 $.d
1) 1) "a"
   2) "b"
   3) "c"
```

제한된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{}, "b":{"a":"a"}, "c":{"a":"a", "b":"bb"}, "d":
{"a":1, "b":"b", "c":{"a":3,"b":4}}, "e":1}'
OK
127.0.0.1:6379> JSON.OBJKEYS k1 .*
1) "a"
127.0.0.1:6379> JSON.OBJKEYS k1 .d
1) "a"
2) "b"
3) "c"
```

JSON.RESP

Valkey 또는 Redis OSS(Redis Serialization Protocol)의 지정된 경로에 JSON 값을 반환합니다. 값이 컨테이너인 경우, 응답은 RESP 배열 또는 중첩 배열입니다.

- JSON null은 RESP Null 대량 문자열에 매핑됩니다.
- JSON 부울 값은 각 RESP 단순 문자열에 매핑됩니다.
- 정수는 RESP 정수에 매핑됩니다.
- 64비트 IEEE 이중 부동 소수점 숫자는 RESP 대량 문자열에 매핑됩니다.

- JSON 문자열은 RESP 대량 문자열에 매핑됩니다.
- JSON 배열은 RESP 배열로 표현됩니다. 여기서 첫 번째 요소는 간단한 문자열 [이고 그 뒤에 배열의 요소가 옵니다.
- JSON 객체는 RESP 배열로 표현됩니다. 여기서 첫 번째 요소는 간단한 문자열 {이고 그 뒤에 각각이 RESP 대량 문자열인 키-값 쌍이 옵니다.

구문

```
JSON.RESP <key> [path]
```

- 키(필수) - JSON 문서 유형의 키입니다.
- 경로(선택 사항) - JSON 경로입니다. 제공하지 않으면 기본적으로 root로 설정됩니다.

반환

경로가 향상된 구문인 경우

- 배열의 배열입니다. 각 배열 요소는 한 경로에 있는 값의 RESP 형식을 나타냅니다.
- 문서 키가 없는 경우 빈 배열입니다.

경로가 제한된 구문인 경우

- 경로에 있는 값의 RESP 형식을 나타내는 배열입니다.
- 문서 키가 없으면 Null입니다.

예

향상된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 .
 '{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"},{"type":"office","number":"646
 555-4567"}],"children":[],"spouse":null}'
OK
```

```
127.0.0.1:6379> JSON.RESP k1 $.address
1) 1) {
   2) 1) "street"
      2) "21 2nd Street"
   3) 1) "city"
      2) "New York"
   4) 1) "state"
      2) "NY"
   5) 1) "zipcode"
      2) "10021-3100"

127.0.0.1:6379> JSON.RESP k1 $.address.*
1) "21 2nd Street"
2) "New York"
3) "NY"
4) "10021-3100"

127.0.0.1:6379> JSON.RESP k1 $.phoneNumbers
1) 1) [
   2) 1) {
      2) 1) "type"
         2) "home"
      3) 1) "number"
         2) "555 555-1234"
   3) 1) {
      2) 1) "type"
         2) "office"
      3) 1) "number"
         2) "555 555-4567"

127.0.0.1:6379> JSON.RESP k1 $.phoneNumbers[*]
1) 1) {
   2) 1) "type"
      2) "home"
   3) 1) "number"
      2) "212 555-1234"
2) 1) {
   2) 1) "type"
      2) "office"
   3) 1) "number"
      2) "555 555-4567"
```

제한된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 .
 '{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"},{"type":"office","number":"646
 555-4567"}],"children":[],"spouse":null}'
OK

127.0.0.1:6379> JSON.RESP k1 .address
1) {
2) 1) "street"
   2) "21 2nd Street"
3) 1) "city"
   2) "New York"
4) 1) "state"
   2) "NY"
5) 1) "zipcode"
   2) "10021-3100"

127.0.0.1:6379> JSON.RESP k1
 1) {
 2) 1) "firstName"
    2) "John"
 3) 1) "lastName"
    2) "Smith"
 4) 1) "age"
    2) (integer) 27
 5) 1) "weight"
    2) "135.25"
 6) 1) "isAlive"
    2) true
 7) 1) "address"
    2) 1) {
       2) 1) "street"
          2) "21 2nd Street"
       3) 1) "city"
          2) "New York"
       4) 1) "state"
          2) "NY"
       5) 1) "zipcode"
          2) "10021-3100"
 8) 1) "phoneNumbers"
```

```
    2) 1) [
       2) 1) {
          2) 1) "type"
             2) "home"
          3) 1) "number"
             2) "212 555-1234"
       3) 1) {
          2) 1) "type"
             2) "office"
          3) 1) "number"
             2) "555 555-4567"
 9) 1) "children"
    2) 1) [
10) 1) "spouse"
    2) (nil)
```

JSON.SET

경로에 JSON 값을 설정합니다.

경로가 객체 멤버를 호출하는 경우

- 상위 요소가 없는 경우, 다음 명령은 NONEXISTENT 오류를 반환합니다.
- 상위 요소가 있지만 객체가 아닌 경우, 명령은 ERROR를 반환합니다.
- 상위 요소가 있고 객체인 경우
  - 멤버가 없으면 상위 객체가 경로의 마지막 하위 객체인 경우에만 새 멤버는 상위 객체에 추가됩니다. 그렇지 않으면 명령은 NONEXISTENT 오류를 반환합니다.
  - 멤버가 있으면 해당 값이 JSON 값으로 대체됩니다.

경로가 배열 인덱스를 호출하는 경우

- 상위 요소가 없는 경우, 다음 명령은 NONEXISTENT 오류를 반환합니다.
- 상위 요소가 있지만 배열이 아닌 경우, 명령은 ERROR를 반환합니다.
- 상위 요소가 있지만 인덱스가 범위를 벗어난 경우, 명령은 OUTOFBOUNDARIES 오류를 반환합니다.
- 상위 요소가 있고 인덱스가 유효하면 요소는 새 JSON 값으로 대체됩니다.

경로가 객체 또는 배열을 호출하는 경우 값(객체 또는 배열)은 새 JSON 값으로 대체됩니다.

구문

```
JSON.SET <key> <path> <json> [NX | XX]
```

[NX | XX] 여기에 [NX | XX] 식별자 0개 또는 1개가 있을 수 있습니다..

- 키(필수) - JSON 문서 유형의 키입니다.
- 경로(필수) - JSON 경로입니다. 새 키의 경우 JSON 경로가 루트 '.'여야 합니다.
- NX(선택 사항) - 경로가 루트이면, 키가 없는 경우에만 값을 설정합니다(예: 새 문서 삽입). 경로가 루트가 아니면 경로가 없는 경우에만 값을 설정합니다.
- XX(선택 사항) - 경로가 루트이면, Redis 키가 있는 경우에만 값을 설정합니다(예: 기존 문서 대체). 경로가 루트가 아니면 경로가 있는 경우에만 값을 설정합니다.

반환

- 성공 시 단순 문자열 '확인'.
- NX 또는 XX 조건이 충족되지 않으면 null입니다.

예

향상된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . '{"a":{"a":1, "b":2, "c":3}}'
OK
127.0.0.1:6379> JSON.SET k1 $.a.* '0'
OK
127.0.0.1:6379> JSON.GET k1
"{\"a\":{\"a\":0,\"b\":0,\"c\":0}}"

127.0.0.1:6379> JSON.SET k2 . '{"a": [1,2,3,4,5]}'
OK
127.0.0.1:6379> JSON.SET k2 $.a[*] '0'
OK
127.0.0.1:6379> JSON.GET k2
"{\"a\":[0,0,0,0,0]}"
```

제한된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . '{"c":{"a":1, "b":2}, "e": [1,2,3,4,5]}'
OK
127.0.0.1:6379> JSON.SET k1 .c.a '0'
OK
127.0.0.1:6379> JSON.GET k1
"{\"c\":{\"a\":0,\"b\":2},\"e\":[1,2,3,4,5]}"
127.0.0.1:6379> JSON.SET k1 .e[-1] '0'
OK
127.0.0.1:6379> JSON.GET k1
"{\"c\":{\"a\":0,\"b\":2},\"e\":[1,2,3,4,0]}"
127.0.0.1:6379> JSON.SET k1 .e[5] '0'
(error) OUTOFBOUNDARIES Array index is out of bounds
```

JSON.STRAPPEND

경로에서 JSON 문자열에 문자열을 추가합니다.

구문

```
JSON.STRAPPEND <key> [path] <json_string>
```

- 키(필수) - JSON 문서 유형의 키입니다.
- 경로(선택 사항) - JSON 경로입니다. 제공하지 않으면 기본적으로 root로 설정됩니다.
- json_string(필수) - 문자열의 JSON 표현입니다. JSON 문자열은 따옴표로 묶어야 합니다. 즉, '"foo"'.

반환

경로가 향상된 구문인 경우

- 각 경로에서 문자열의 새 길이를 나타내는 정수 배열입니다.
- 경로의 값이 문자열이 아닌 경우 해당 반환 값은 null입니다.
- 입력 json 인수가 유효한 JSON 문자열이 아닌 경우, `SYNTAXERR` 오류가 발생합니다.
- 경로가 존재하지 않는 경우, `NONEXISTENT` 오류가 발생합니다.

경로가 제한된 구문인 경우

- 정수, 문자열의 새 길이입니다.

- 여러 문자열 값을 선택한 경우 명령은 마지막으로 업데이트된 문자열의 새 길이를 반환합니다.
- 경로의 값이 문자열이 아닌 경우 `WRONGTYPE` 오류가 발생합니다.
- 입력 json 인수가 유효한 JSON 문자열이 아닌 경우 `WRONGTYPE` 오류가 발생합니다.
- `NONEXISTENT`경로가 없는 경우 오류가 발생합니다.

예

향상된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"bb"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.STRAPPEND k1 $.a.a '"a"'
1) (integer) 2
127.0.0.1:6379> JSON.STRAPPEND k1 $.a.* '"a"'
1) (integer) 3
127.0.0.1:6379> JSON.STRAPPEND k1 $.b.* '"a"'
1) (integer) 2
2) (nil)
127.0.0.1:6379> JSON.STRAPPEND k1 $.c.* '"a"'
1) (integer) 2
2) (integer) 3
127.0.0.1:6379> JSON.STRAPPEND k1 $.c.b '"a"'
1) (integer) 4
127.0.0.1:6379> JSON.STRAPPEND k1 $.d.* '"a"'
1) (nil)
2) (integer) 2
3) (nil)
```

제한된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"bb"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.STRAPPEND k1 .a.a '"a"'
(integer) 2
127.0.0.1:6379> JSON.STRAPPEND k1 .a.* '"a"'
(integer) 3
127.0.0.1:6379> JSON.STRAPPEND k1 .b.* '"a"'
(integer) 2
```

```
127.0.0.1:6379> JSON.STRAPPEND k1 .c.* '"a"'
(integer) 3
127.0.0.1:6379> JSON.STRAPPEND k1 .c.b '"a"'
(integer) 4
127.0.0.1:6379> JSON.STRAPPEND k1 .d.* '"a"'
(integer) 2
```

JSON.STRLEN

경로에서 JSON 문자열 값의 길이를 얻습니다.

구문

```
JSON.STRLEN <key> [path]
```

- 키(필수) - JSON 문서 유형의 키입니다.
- 경로(선택 사항) - JSON 경로입니다. 제공하지 않으면 기본적으로 root로 설정됩니다.

반환

경로가 향상된 구문인 경우

- 각 경로에서 문자열 값의 길이를 나타내는 정수 배열입니다.
- 값이 문자열이 아닌 경우 해당 반환 값은 null입니다.
- 문서 키가 없으면 null입니다.

경로가 제한된 구문인 경우

- 정수, 문자열의 길이입니다.
- 여러 문자열 값을 선택한 경우 명령은 첫 번째 문자열의 길이를 반환합니다.
- 경로의 값이 문자열이 아닌 경우 `WRONGTYPE` 오류가 발생합니다.
- `NONEXISTENT`경로가 존재하지 않는 경우 오류가 발생합니다.
- 문서 키가 없으면 Null입니다.

예

향상된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"bb"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.STRLEN k1 $.a.a
1) (integer) 1
127.0.0.1:6379> JSON.STRLEN k1 $.a.*
1) (integer) 1
127.0.0.1:6379> JSON.STRLEN k1 $.c.*
1) (integer) 1
2) (integer) 2
127.0.0.1:6379> JSON.STRLEN k1 $.c.b
1) (integer) 2
127.0.0.1:6379> JSON.STRLEN k1 $.d.*
1) (nil)
2) (integer) 1
3) (nil)
```

제한된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"bb"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.STRLEN k1 .a.a
(integer) 1
127.0.0.1:6379> JSON.STRLEN k1 .a.*
(integer) 1
127.0.0.1:6379> JSON.STRLEN k1 .c.*
(integer) 1
127.0.0.1:6379> JSON.STRLEN k1 .c.b
(integer) 2
127.0.0.1:6379> JSON.STRLEN k1 .d.*
(integer) 1
```

JSON.TOGGLE

경로에서 참과 거짓 사이의 부울 값을 토글합니다.

구문

```
JSON.TOGGLE <key> [path]
```

- 키(필수) - JSON 문서 유형의 키입니다.
- 경로(선택 사항) - JSON 경로입니다. 제공하지 않으면 기본적으로 root로 설정됩니다.

반환

경로가 향상된 구문인 경우

- 각 경로에서 결과 부울 값을 나타내는 정수 배열(0 - 거짓, 1 - 참)입니다.
- 값이 배열이 부울 값이 아닌 경우, 해당 반환 값은 null입니다.
- 문서 키가 없는 경우 `NONEXISTENT`입니다.

경로가 제한된 구문인 경우

- 결과 부울 값을 나타내는 문자열('참'/ '거짓')입니다.
- 문서 키가 없는 경우 `NONEXISTENT`입니다.
- 경로의 값이 부울 값이 아닌 경우, `WRONGTYPE` 오류가 발생합니다.

예

향상된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . '{"a":true, "b":false, "c":1, "d":null, "e":"foo", "f":
[], "g":{}}'
OK
127.0.0.1:6379> JSON.TOGGLE k1 $.*
1) (integer) 0
2) (integer) 1
3) (nil)
4) (nil)
5) (nil)
6) (nil)
7) (nil)
127.0.0.1:6379> JSON.TOGGLE k1 $.*
1) (integer) 1
2) (integer) 0
3) (nil)
```

```
4) (nil)
5) (nil)
6) (nil)
7) (nil)
```

제한된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . true
OK
127.0.0.1:6379> JSON.TOGGLE k1
"false"
127.0.0.1:6379> JSON.TOGGLE k1
"true"

127.0.0.1:6379> JSON.SET k2 . '{"isAvailable": false}'
OK
127.0.0.1:6379> JSON.TOGGLE k2 .isAvailable
"true"
127.0.0.1:6379> JSON.TOGGLE k2 .isAvailable
"false"
```

JSON.TYPE

지정된 경로의 값 유형을 보고합니다.

구문

```
JSON.TYPE <key> [path]
```

- 키(필수) - JSON 문서 유형의 키입니다.
- 경로(선택 사항) - JSON 경로입니다. 제공하지 않으면 기본적으로 root로 설정됩니다.

반환

경로가 향상된 구문인 경우

- 각 경로에서 값 유형을 나타내는 문자열 배열입니다. 유형은 {"null', '부울', '문자열', '숫자', '정수', '개체'및 '배열'} 중의 하나입니다.

- 경로가 없는 경우 해당 값은 null입니다.
- 문서 키가 없는 경우 빈 배열입니다.

경로가 제한된 구문인 경우

- 문자열, 값 유형
- 문서 키가 없으면 null입니다.
- JSON 경로가 잘못되었거나 없는 경우 null입니다.

예

향상된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 . '[1, 2.3, "foo", true, null, {}, []]'
OK
127.0.0.1:6379> JSON.TYPE k1 $[*]
1) integer
2) number
3) string
4) boolean
5) null
6) object
7) array
```

제한된 경로 구문.

```
127.0.0.1:6379> JSON.SET k1 .
 '{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"},{"type":"office","number":"646
 555-4567"}],"children":[],"spouse":null}'
OK
127.0.0.1:6379> JSON.TYPE k1
object
127.0.0.1:6379> JSON.TYPE k1 .children
array
127.0.0.1:6379> JSON.TYPE k1 .firstName
string
```

```
127.0.0.1:6379> JSON.TYPE k1 .age
integer
127.0.0.1:6379> JSON.TYPE k1 .weight
number
127.0.0.1:6379> JSON.TYPE k1 .isAlive
boolean
127.0.0.1:6379> JSON.TYPE k1 .spouse
null
```

MemoryDB 리소스 태그 지정

클러스터 및 기타 MemoryDB 리소스 관리를 돕기 위해 태그 형식으로 각 리소스에 고유한 메타데이터를 할당할 수 있습니다. 태그를 사용하면 용도, 소유자 또는 환경을 기준으로 하는 등 AWS 리소스를 다양한 방식으로 분류할 수 있습니다. 이 기능은 동일 유형의 리소스가 많을 때 유용합니다. 지정한 태그에 따라 특정 리소스를 빠르게 식별할 수 있습니다. 이 주제에서는 태그를 설명하고 태그를 생성하는 방법을 보여줍니다.

> **Warning**
> 민감한 데이터를 태그에 포함하지 않는 것이 가장 좋습니다.

태그 기본 사항

태그는 AWS 리소스에 할당하는 레이블입니다. 각 태그는 사용자가 정의하는 키와 선택적 값으로 구성됩니다. 태그를 사용하면 용도, 소유자 등을 기준으로 AWS 리소스를 다양한 방식으로 분류할 수 있습니다. 예를 들어, 계정의 MemoryDB 클러스터에 대해 각 클러스터의 소유자와 사용자 그룹을 추적하는데 도움이 되는 태그 세트를 정의할 수 있습니다.

각 리소스 유형에 대한 요건을 충족하는 태그 키 세트를 고안하는 것이 좋습니다. 일관된 태그 키 세트를 사용하면 리소스를 보다 쉽게 관리할 수 있습니다. 추가하는 태그에 따라 리소스를 검색하고 필터링할 수 있습니다. 효과적인 리소스 태그 지정 전략을 구현하는 방법에 대한 자세한 정보는 AWS 백서 태그 지정 모범 사례를 참조하세요.

태그는 MemoryDB에는 의미가 없으며 엄격하게 문자열로 해석됩니다. 또한 태그는 리소스에 자동으로 배정되지 않습니다. 태그 키와 값을 편집할 수 있으며 언제든지 리소스에서 태그를 제거할 수 있습니다. 태그의 값을 `null`로 설정할 수 있습니다. 해당 리소스에 대해 키가 기존 태그와 동일한 태그를 추가하는 경우 새 값이 이전 값을 덮어씁니다. 리소스를 삭제하면 리소스 태그도 삭제됩니다.

AWS Management 콘솔, AWS CLI, MemoryDB API를 사용하여 태그 관련 작업을 수행할 수 있습니다.

IAM을 사용하는 경우 AWS 계정에서 태그를 생성, 편집 또는 삭제할 수 있는 권한이 있는 사용자를 제어할 수 있습니다. 자세한 내용은 리소스 수준 권한 섹션을 참조하세요.

태그 지정이 가능한 리소스

계정에 이미 존재하는 대부분의 MemoryDB 리소스에 태그를 지정할 수 있습니다. 아래의 표에는 태그 지정을 지원하는 리소스가 나와 있습니다. AWS Management 콘솔을 사용 중인 경우 Tag Editor를 사용하여 리소스에 태그를 적용할 수 있습니다. 일부 리소스 화면을 사용하면 리소스를 생성할 때 리소스에 대해 태그를 지정할 수 있습니다. 예를 들어 Name의 키가 있는 태그와 지정하는 값이 있습니다. 대부분의 경우, 콘솔은 리소스 생성 직후(리소스 생성 중이 아니라) 태그를 적용합니다. 콘솔은 Name 태그에 따라 리소스를 조직할 수 있지만 이 태그는 MemoryDB 서비스에 대한 의미가 없습니다.

또한 일부 리소스 생성 작업에서는 리소스 생성 시 리소스의 태그를 지정할 수 있습니다. 리소스 생성 도중 태그를 적용할 수 없는 경우, 리소스 생성 프로세스가 롤백됩니다. 이는 태그를 사용하여 리소스가 생성되거나 아예 리소스가 생성되지 않도록 하고 언제든 태그 지정되지 않은 리소스가 남지 않게 합니다. 생성 시 리소스에 태그를 지정하면 리소스 생성 후 사용자 지정 태그 지정 스크립트를 실행할 필요가 없습니다.

Amazon MemoryDB API, AWS CLI 또는 AWS SDK를 사용 중인 경우 관련 MemoryDB API 작업의 `Tags` 파라미터를 사용하여 태그를 적용할 수 있습니다. 스크립트는 다음과 같습니다.

- `CreateCluster`
- `CopySnapshot`
- `CreateParameterGroup`
- `CreateSubnetGroup`
- `CreateSnapshot`
- `CreateACL`
- `CreateUser`
- `CreateMultiRegionCluster`

다음 표는 태그를 지정할 수 있는 MemoryDB 리소스와 MemoryDB API, AWS CLI 또는 AWS SDK를 사용하여 생성 시 태그를 지정할 수 있는 리소스를 설명합니다.

MemoryDB 리소스 태그 지정 지원

태그 지원	생성 시 태그 지정 지원
예	예
예	예
예	예
예	예
예	예
예	예
예	예

생성 시 태그를 지원하는 MemoryDB API 작업에 IAM 정책의 태그 기반 리소스 수준 권한을 적용하여 생성 시 리소스에 태그를 지정할 수 있는 사용자와 그룹을 세밀하게 제어할 수 있습니다. 리소스를 생성하면 태그가 즉시 적용되기 때문에 생성 단계부터 리소스를 적절하게 보호할 수 있습니다. 따라서 태그를 기반으로 리소스 사용을 제어하는 리소스 수준 권한이 즉시 발효됩니다. 이에 따라 더욱 정확한 리소스 추적 및 보고가 가능합니다. 새 리소스에서 태그 지정 사용을 적용하고 리소스에서 어떤 태그 키와 값이 설정되는지 제어할 수 있습니다.

자세한 내용은 [리소스에 태그 지정 예제] 섹션을 참조하세요.

결제를 위한 리소스 태그 지정에 대한 자세한 내용은 [비용 할당 태그를 사용하여 비용 모니터링] 섹션을 참조하세요.

클러스터 및 스냅샷과 Multi-Region 클러스터에 태그 지정

태그 지정에는 요청 작업의 일부로 다음 규칙이 적용됩니다.

- CreateCluster :
  - `--cluster-name`이 제공된 경우:

    태그가 요청에 포함되어 있으면 클러스터에 태그가 지정됩니다.
  - `--snapshot-name`이 제공된 경우:

    태그가 요청에 포함되어 있으면 클러스터에는 해당 태그로만 태그가 지정됩니다. 요청에 태그가 포함되어 있지 않은 경우 스냅샷 태그가 클러스터에 추가됩니다.
- CreateSnapshot:
  - `--cluster-name`이 제공된 경우:

    태그가 요청에 포함되어 있으면 요청 태그만 스냅샷에 추가됩니다. 요청에 태그가 포함되어 있지 않은 경우 클러스터 태그가 스냅샷에 추가됩니다.
  - 자동 스냅샷의 경우:

    태그가 클러스터 태그에서 전파됩니다.
- CopySnapshot:

  태그가 요청에 포함되어 있으면 요청 태그만 스냅샷에 추가됩니다. 요청에 태그가 포함되어 있지 않은 경우 소스 스냅샷 태그가 복사된 스냅샷에 추가됩니다.
- TagResource 및 UntagResource :

  리소스에서 태그가 추가/제거됩니다.

다중 리전 클러스터 태그 지정

MemoryDB 다중 리전 클러스터는 글로벌 리소스입니다. 따라서 MemoryDB Multi-Region이 지원되는 특정 리전에서 관련 APIs를 호출하여 다중 리전 클러스터에서 태그를 지정, 수정 또는 나열할 수 있습니다. 리전 지원에 대한 자세한 내용은 사전 조건 및 제한 사항을 참조하세요.

다중 리전 클러스터의 태그는 리전 클러스터의 태그와 독립적입니다. 다중 리전 클러스터에서 다양한 태그 세트를 지정할 수 있으며 리전 클러스터가 포함되어 있습니다. 이러한 태그 간에는 계층적 연결이 없으며 이러한 리소스 유형 간의 계층 구조를 통해 복사되지 않습니다.

`TagResource` 및 `UntagResource` APIs를 통해 태그를 추가하거나 제거할 때 태그가 특히 다중 리전 클러스터에 대해 일관되게 유지되므로 ListTags API 응답에 최신 유효 태그가 즉시 표시되지 않을 수 있습니다.

태그 제한

태그에 적용되는 기본 제한은 다음과 같습니다.

- 리소스당 최대 태그 수 – 50개
- 각 리소스에 대해 각 태그 키는 고유하며 하나의 값만 가질 수 있습니다.
- 최대 키 길이는 유니코드 문자(UTF-8) 128자입니다.
- 최대 값 길이는 유니코드 문자(UTF-8) 256자입니다.
- MemoryDB는 태그에 모든 문자를 사용할 수 있지만, 다른 서비스에는 제한이 적용될 수 있습니다. 서비스에서 허용되는 문자는 UTF-8로 표현할 수 있는 문자, 숫자 및 공백과 특수 문자 + - = . _ : / @입니다.
- 태그 키와 값은 대/소문자를 구분합니다.
- `aws:` 접두사는 AWS용으로 예약되어 있습니다. 태그에 이 접두사가 있는 태그 키가 있는 경우 태그의 키 또는 값을 편집하거나 삭제할 수 없습니다. `aws:` 접두사가 지정된 태그는 리소스당 태그 수 제한에 포함되지 않습니다.

태그에만 기초하여 리소스를 종료, 중지 또는 삭제할 수 없습니다. 리소스 식별자를 지정해야 합니다. 예를 들어 `DeleteMe`라는 태그 키로 태그를 지정한 스냅샷을 삭제하려면 해당 스냅샷의 리소스 식별자(예: `DeleteSnapshot`)를 지정하여 `snap-1234567890abcdef0` 작업을 사용해야 합니다.

태그를 지정할 수 있는 MemoryDB 리소스에 대한 자세한 내용은 [태그 지정이 가능한 리소스](#) 섹션을 참조하세요.

리소스에 태그 지정 예제

- 새 클러스터에 태그 추가

```
aws memorydb tag-resource \
--resource-arn arn:aws:memorydb:us-east-1:111111222233:cluster/my-cluster \
--tags Key="project",Value="XYZ" Key="memorydb",Value="Service"
```

- 태그를 사용하여 클러스터 생성

```
aws memorydb create-cluster \
```

```
--cluster-name testing-tags \
--description cluster-test \
--subnet-group-name test \
--node-type db.r6g.large \
--acl-name open-access \
--tags Key="project",Value="XYZ" Key="memorydb",Value="Service"
```

- 태그를 사용하여 스냅샷 생성

  이 경우 요청에 태그를 추가하면 클러스터에 태그가 포함되어 있더라도 스냅샷은 요청 태그만 받습니다.

```
aws memorydb create-snapshot \
--cluster-name testing-tags \
--snapshot-name bkp-testing-tags-mycluster \
--tags Key="work",Value="foo"
```

비용 할당 태그를 사용하여 비용 모니터링

MemoryDB에서 리소스에 비용 할당 태그를 추가하면 인보이스 비용을 리소스 태그 값으로 그룹화하여 비용을 추적할 수 있습니다.

MemoryDB 비용 할당 태그는 사용자가 정의하고 MemoryDB 리소스에 연결하는 키 값 페어입니다. 키와 값은 대/소문자를 구분합니다. 태그 키를 사용하여 범주를 정의할 수 있으며 태그 값은 해당 범주의 항목일 수 있습니다. 예를 들어, `CostCenter`의 태그 키와 `10010`의 태그 값을 정의하여 리소스가 10010 코스트 센터에 할당됨을 나타냅니다. `Environment`와 같은 키 및 `test` 또는 `production`과 같은 값을 사용하여 태그로 리소스를 테스트나 프로덕션에 사용되는 것으로 지정할 수도 있습니다. 리소스 관련 비용을 보다 쉽게 추적할 수 있도록 하기 위해 일관된 태그 키 세트를 사용하는 것이 좋습니다.

비용 할당 태그를 사용하여 자신만의 비용 구조를 반영하도록 AWS 청구서를 구성합니다. 이렇게 하려면 가입하여 태그 키 값이 포함된 AWS 계정 청구서를 가져옵니다. 그런 다음 같은 태그 키 값을 가진 리소스에 따라 결제 정보를 구성하여 리소스 비용의 합을 볼 수 있습니다. 예를 들어, 특정 애플리케이션 이름으로 여러 리소스에 태그를 지정한 다음 결제 정보를 구성하여 여러 서비스에 걸친 해당 애플리케이션의 총 비용을 볼 수 있습니다.

또한 태그를 결합하여 보다 세부적인 수준으로 비용을 추적할 수 있습니다. 예를 들어, 리전별 서비스 비용을 추적하려면 `Service` 및 `Region` 태그 키를 사용할 수 있습니다. 하나의 리소스에서 `MemoryDB` 및 `Asia Pacific (Singapore)` 값이 있을 수 있으며 다른 리소스에서 `MemoryDB` 및

`Europe (Frankfurt)` 값이 있을 수 있습니다. 리전별로 구분된 총 MemoryDB 비용을 볼 수 있습니다. 자세한 내용은 AWS Billing 사용 설명서의 비용 할당 태그 사용을 참조하세요.

MemoryDB 클러스터에 MemoryDB 비용 할당 태그를 추가할 수 있습니다. 태그를 추가, 나열, 수정, 복사 또는 제거할 때 이 작업은 지정된 클러스터에만 적용됩니다.

MemoryDB 비용 할당 태그 특성

- 비용 할당 태그는 CLI 및 API 작업에서 ARN으로 지정된 MemoryDB 리소스에 적용됩니다. 리소스 유형은 "클러스터"입니다.

  ARN 형식: `arn:aws:memorydb:`*<region>*`:`*<customer-id>*`:`*<resource-type>*`/`*<resource-name>*

  샘플 ARN: `arn:aws:memorydb:us-east-1:1234567890:cluster/my-cluster`
- 태그 키는 태그의 필수 이름입니다. 키의 문자열 값은 1~128자(유니코드 문자) 사이가 될 수 있으며 `aws:`로 시작할 수 없습니다. 문자열에는 유니코드 문자, 숫자, 공백, 밑줄(_), 마침표(.), 콜론(:), 백슬래시(\ ), 등호(=), 더하기 기호(+), 하이픈(-), at 기호(@) 집합만 포함될 수 있습니다.
- 태그 값은 태그의 선택적 값입니다. 값의 문자열 값은 1~256자(유니코드 문자) 사이가 될 수 있으며 `aws:`로 시작할 수 없습니다. 문자열에는 유니코드 문자, 숫자, 공백, 밑줄(_), 마침표(.), 콜론(:), 백슬래시(\ ), 등호(=), 더하기 기호(+), 하이픈(-), at 기호(@) 집합만 포함될 수 있습니다.
- MemoryDB 리소스는 최대 50개의 태그를 보유할 수 있습니다.
- 태그 세트의 값이 고유하지 않습니다. 예를 들어, 두 키 `Service`와 `Application`에 `MemoryDB` 값이 있는 태그 세트가 있을 수 있습니다.

AWS는 태그에 의미론적 의미를 적용하지 않습니다. 태그는 엄격히 문자열로 해석됩니다. AWS에서는 MemoryDB 리소스에 어떠한 태그도 자동으로 설정하지 않습니다.

AWS CLI를 사용하여 비용 할당 태그 관리

AWS CLI를 사용하여 비용 할당 태그를 추가, 수정 또는 제거할 수 있습니다.

샘플 arn: `arn:aws:memorydb:us-east-1:1234567890:cluster/my-cluster`

주제

- AWS CLI를 사용하여 태그 나열
- AWS CLI를 사용하여 태그 추가
- AWS CLI를 사용하여 태그 수정

- [AWS CLI를 사용하여 태그 제거](#)

AWS CLI를 사용하여 태그 나열

list-tags 작업을 사용하여 기존 MemoryDB 리소스에서 태그를 나열하는 데 AWS CLI을(를) 사용할 수 있습니다.

다음 코드는 AWS CLI를 사용하여 us-west-1 리전에 있는 MemoryDB 클러스터 `my-cluster`의 태그를 나열합니다.

Linux, macOS, Unix의 경우:

```
aws memorydb list-tags \
  --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster
```

Windows의 경우:

```
aws memorydb list-tags ^
  --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster
```

이 작업의 출력은 다음 리소스의 모든 태그 목록과 유사합니다.

```
{
   "TagList": [
      {
         "Value": "10110",
         "Key": "CostCenter"
      },
      {
         "Value": "EC2",
         "Key": "Service"
      }
   ]
}
```

리소스에 태그가 없으면 출력은 빈 TagList가 됩니다.

```
{
   "TagList": []
}
```

자세한 내용은 for MemoryDB 목록 태그를 참조하세요AWS CLI.

AWS CLI를 사용하여 태그 추가

tag-resource CLI 작업을 사용하여 기존 리소스에 태그를 추가하는 데 AWS CLI를 사용할 수 있습니다. 리소스에 태그 키가 없으면 키와 값이 리소스에 추가됩니다. 리소스에 이미 키가 있는 경우 해당 키와 연결된 값이 새 값으로 업데이트됩니다.

다음 코드는 AWS CLI을(를) 사용하여 us-west-1 리전에 있는 클러스터 `my-cluster`에 각각 `memorydb` 및 `us-east-1` 값을 갖는 `Service` 및 `Region` 키를 추가합니다.

Linux, macOS, Unix의 경우:

```
aws memorydb tag-resource \
 --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster \
 --tags Key=Service,Value=memorydb \
        Key=Region,Value=us-east-1
```

Windows의 경우:

```
aws memorydb tag-resource ^
 --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster ^
 --tags Key=Service,Value=memorydb ^
        Key=Region,Value=us-east-1
```

이 작업의 출력은 다음 작업 후 리소스의 모든 태그 목록과 유사합니다.

```
{
   "TagList": [
      {
         "Value": "memorydb",
         "Key": "Service"
      },
      {
         "Value": "us-east-1",
         "Key": "Region"
      }
   ]
}
```

자세한 내용은 MemoryDB tag-resource에 대해 AWS CLI을 참조하세요.

create-cluster 작업을 사용하여 새 클러스터를 생성할 때 AWS CLI을(를) 사용하여 클러스터에 태그를 추가할 수도 있습니다.

AWS CLI를 사용하여 태그 수정

AWS CLI을(를) 사용하여 MemoryDB 클러스터의 태그를 수정할 수 있습니다.

태그를 수정하려면

- tag-resource를 사용하여 새 태그 및 값을 추가하거나 기존 태그에 연결된 값을 변경합니다.
- untag-resource를 사용하여 리소스에서 지정된 태그를 제거합니다.

두 작업 중 하나의 출력은 지정된 클러스터의 태그와 이 태그의 값이 나열된 목록입니다.

AWS CLI를 사용하여 태그 제거

untag-resource 작업을 사용하여 기존 MemoryDB 클러스터에서 태그를 제거하는 데 AWS CLI을(를) 사용할 수 있습니다.

다음 코드에서는 AWS CLI을(를) 사용하여 us-west-1 리전에 있는 클러스터 `my-cluster`에서 `Service` 및 `Region` 키를 갖는 태그를 제거합니다.

Linux, macOS, Unix의 경우:

```
aws memorydb untag-resource \
 --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster \
 --tag-keys Region Service
```

Windows의 경우:

```
aws memorydb untag-resource ^
 --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster ^
 --tag-keys Region Service
```

이 작업의 출력은 다음 작업 후 리소스의 모든 태그 목록과 유사합니다.

```
{
   "TagList": []
}
```

자세한 내용은 MemoryDB untag-resource리소스에 대한 AWS CLI을(를) 참조하세요.

MemoryDB API를 사용하여 비용 할당 태그 관리

MemoryDB API를 사용하여 비용 할당 태그를 추가, 수정 또는 제거할 수 있습니다.

비용 할당 태그는 MemoryDB용 클러스터에 적용됩니다. 태그를 지정할 클러스터는 Amazon 리소스 이름(ARN)을 사용해 지정합니다.

샘플 arn: `arn:aws:memorydb:us-east-1:1234567890:cluster/my-cluster`

주제

- MemoryDB API를 사용하여 태그 나열
- MemoryDB API를 사용하여 태그 추가
- MemoryDB API를 사용하여 태그 수정
- MemoryDB API를 사용하여 태그 제거

MemoryDB API를 사용하여 태그 나열

MemoryDB API를 통해 ListTags 작업을 사용하여 기존 리소스에서 태그를 나열할 수 있습니다.

다음 코드는 MemoryDB API를 사용하여 us-west-1 리전에 있는 리소스 `my-cluster`의 태그를 나열합니다.

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=ListTags
   &ResourceArn=arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Version=2021-01-01
   &Timestamp=20210802T192317Z
   &X-Amz-Credential=<credential>
```

MemoryDB API를 사용하여 태그 추가

TagResource작업을 사용하여 기존 MemoryDB 클러스터에 태그를 추가하는 데 MemoryDB API를 사용할 수 있습니다. 리소스에 태그 키가 없으면 키와 값이 리소스에 추가됩니다. 리소스에 이미 키가 있는 경우 해당 키와 연결된 값이 새 값으로 업데이트됩니다.

다음 코드는 MemoryDB API를 사용하여 us-east-1 리전의 리소스 `my-cluster`에 각각 `memorydb` 및 `us-east-1` 값을 갖는 `Service` 및 `Region` 키를 추가합니다.

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=TagResource
   &ResourceArn=arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Tags.member.1.Key=Service
   &Tags.member.1.Value=memorydb
   &Tags.member.2.Key=Region
   &Tags.member.2.Value=us-east-1
   &Version=2021-01-01
   &Timestamp=20210802T192317Z
   &X-Amz-Credential=<credential>
```

자세한 내용은 TagResource를 참조하세요.

MemoryDB API를 사용하여 태그 수정

MemoryDB API를 사용하여 MemoryDB 클러스터의 태그를 수정할 수 있습니다.

태그 값을 수정하는 방법

- TagResource 작업을 사용하여 새 태그와 값을 추가하거나 기존 태그의 값을 변경합니다.
- 리소스에서 태그를 제거하려면 UntagResource를 사용합니다.

두 작업 중 하나의 출력은 지정된 리소스의 태그 목록과 값입니다.

MemoryDB API를 사용하여 태그 제거

UntagResource 작업을 사용하여 기존 MemoryDB 클러스터에서 태그를 제거하는 데 MemoryDB API를 사용할 수 있습니다.

다음 코드는 MemoryDB API를 사용하여 us-west-2 리전에 있는 클러스터 `my-cluster`에서 `Service` 및 `Region` 키를 갖는 태그를 제거합니다.

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=UntagResource
   &ResourceArn=arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster
   &SignatureVersion=4
```

```
    &SignatureMethod=HmacSHA256
    &TagKeys.member.1=Service
    &TagKeys.member.2=Region
    &Version=2021-01-01
    &Timestamp=20210802T192317Z
    &X-Amz-Credential=<credential>
```

유지 관리 관리 중

모든 클러스터에는 시스템 변경 내용이 적용되는 주 단위 유지 관리 기간이 있습니다. 캐시 클러스터 생성 또는 수정할 때 원하는 유지 관리 기간을 지정하지 않으면 MemoryDB가 임의로 선택한 요일에 리전의 유지 관리 기간 내에서 60분의 유지 관리 기간을 지정합니다.

리전별로 8시간 블록 시간 중에서 60분 유지 관리 시간이 임의로 선택됩니다. 다음 표는 기본 유지 관리 기간이 할당된 각 리전별 시간 블록 목록입니다. 리전의 유지 관리 기간 블록 외부에서 원하는 유지 관리 기간을 선택할 수 있습니다.

리전 코드	리전 이름	리전 유지 관리 기간
ap-northeast-1	Asia Pacific (Tokyo) Region	13:00~21:00 UTC
ap-northeast-2	Asia Pacific (Seoul) Region	12:00~20:00 UTC
ap-south-1	Asia Pacific (Mumbai) Region	17:30~1:30 UTC
ap-southeast-1	Asia Pacific (Singapore) Region	14:00~22:00 UTC
ap-east-1	Asia Pacific (Hong Kong) Region	13:00~21:00 UTC
ap-southeast-2	아시아 태평양(시드니) 리전	12:00~20:00 UTC
cn-north-1	중국(베이징) 리전	14:00~22:00 UTC
cn-northwest-1	중국(닝샤) 리전	14:00~22:00 UTC
eu-west-3	EU(파리) 리전	23:59~07:29 UTC
eu-central-1	Europe (Frankfurt) Region	23:00~07:00 UTC
eu-west-1	Europe (Ireland) Region	22:00~06:00 UTC

리전 코드	리전 이름	리전 유지 관리 기간
eu-west-2	Europe (London) Region	23:00~07:00 UTC
sa-east-1	남아메리카(상파울루) 리전	01:00~09:00 UTC
ca-central-1	캐나다(중부) 리전	03:00~11:00 UTC
us-east-1	리전 - 미국 동부(버지니아 북부)	03:00~11:00 UTC
us-east-1	US East (Ohio) Region	04:00~12:00 UTC
us-west-1	US West (N. California) Region	06:00~14:00 UTC
us-west-2	US West (Oregon) Region	06:00~14:00 UTC

클러스터의 유지 관리 기간 변경

유지 관리 기간은 사용률이 가장 낮은 시간에 할당되어야 하므로 수시로 수정되어야 할 수 있습니다. 클러스터를 수정하여 요청한 유지 관리 활동이 이루어지는 기간을 최대 24시간까지 지정할 수 있습니다. 이 시간 동안 사용자가 요청한 지연된 또는 대기 중인 클러스터 수정이 발생합니다.

추가 정보

유지 관리 기간 및 노드 대체에 대한 자세한 내용은 다음을 참조하세요.

- 노드 교체 - 노드 교체 관리
- MemoryDB 클러스터 수정 - 클러스터의 유지 관리 기간 변경

모범 사례

다음은 MemoryDB에 대한 권장 모범 사례입니다. 다음 모범 사례를 준수하면 클러스터의 성능과 안정성이 향상됩니다.

주제

- MemoryDB의 복원력
- 모범 사례: Pub/Sub 및 향상된 I/O 멀티플렉싱
- 모범 사례: 온라인 클러스터 크기 조정

MemoryDB의 복원력

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다. AWS 리전은 물리적으로 분리되고 격리된 다수의 가용 리전을 제공하며 이러한 가용 리전은 짧은 지연 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워크에 연결되어 있습니다. 가용 영역을 사용하면 중단 없이 가용 영역 간에 자동으로 장애 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

AWS 글로벌 인프라 뿐만 아니라 MemoryDB도 데이터 복원력과 스냅샷 요구 사항을 지원하는 다양한 기능을 제공합니다.

주제

- 장애 완화

장애 완화

MemoryDB 구현을 계획할 때는 장애가 애플리케이션 및 데이터에 미치는 영향을 최소화하도록 계획해야 합니다. 이 섹션의 항목은 애플리케이션 및 데이터를 장애로부터 보호하기 위해 취할 수 있는 접근 방식을 다룹니다.

장애 완화: MemoryDB 클러스터

MemoryDB 클러스터는 애플리케이션이 읽고 쓸 수 있는 단일 프라이머리 노드와 0~5개의 읽기 전용 복제본 노드로 구성되어 있습니다. 하지만 고가용성을 위해 최소 1개의 복제본을 사용하는 것이 좋습니다. 프라이머리 노드에 데이터가 작성될 때마다 트랜잭션 로그에 유지되고 복제본 노드에서도 비동기식으로 업데이트됩니다.

읽기 전용 복제본 장애의 경우

1. MemoryDB는 장애가 있는 복제본을 감지합니다.
2. MemoryDB는 장애가 있는 노드를 오프라인 상태로 전환합니다.
3. MemoryDB는 동일한 AZ에서 대체 노드를 시작하고 프로비저닝합니다.
4. 새 노드는 트랜잭션 로그와 동기화됩니다.

이 기간 동안 애플리케이션에서는 다른 노드를 사용하여 계속 읽고 쓸 수 있습니다.

MemoryDB 다중 AZ

MemoryDB 클러스터에서 다중 AZ가 활성화되면 장애가 발생한 프라이머리 클러스터가 감지되어 자동으로 교체됩니다.

1. MemoryDB가 프라이머리 노드 장애를 감지합니다.
2. MemoryDB는 장애가 발생한 기본 복제본과 일관성이 있는지 확인한 후 복제본으로 장애 조치합니다.
3. MemoryDB는 장애가 있는는 프라이머리 노드의 AZ에서 읽기 전용 복제본을 실행합니다.
4. 새 노드는 트랜잭션 로그와 동기화됩니다.

복제본 노드에 장애 조치하는 것은 일반적으로 새 기본 노드를 생성하고 프로비저닝하는 것보다 빠릅니다. 즉, 애플리케이션은 더 빠르게 프라이머리 노드에 대한 쓰기를 재개할 수 있습니다.

자세한 내용은 [다중 AZ로 MemoryDB의 가동 중지 시간 최소화](#) 섹션을 참조하세요.

모범 사례: Pub/Sub 및 향상된 I/O 멀티플렉싱

Valkey 또는 Redis OSS 버전 7 이상을 사용하는 경우 샤딩된 Pub/Sub를 사용하는 것이 좋습니다. 또한 향상된 I/O 멀티플렉싱을 사용하여 처리량과 지연 시간을 개선할 수 있습니다. 향상된 I/O 멀티플렉싱은 Valkey 또는 Redis OSS 버전 7 이상을 사용할 때 자동으로 제공되며 클라이언트를 변경할 필요가 없습니다. 여러 클라이언트 연결로 처리량이 제한되는 Pub/Sub 워크로드에 적합합니다.

모범 사례: 온라인 클러스터 크기 조정

리샤딩에는 클러스터에(서) 샤드 또는 노드를 추가 및 제거하고 샤드 간에 키 공간을 재분산하는 작업이 수반됩니다. 따라서 클러스터에 대한 로드, 메모리 사용률 및 데이터의 전체 크기 등과 같은 여러 가지 요인이 리샤딩 작업에 영향을 미칩니다. 최상의 성능을 구현하기 위해서는 균일한 워크로드 패턴 분산을 위한 전반적인 클러스터 모범 사례를 따르는 것이 좋습니다. 또한 다음 단계를 수행하는 것이 좋습니다.

리샤딩을 시작하기 전에 다음 작업을 수행하는 것이 좋습니다.

- 애플리케이션 테스트 - 가능한 경우, 준비 환경에서 리샤딩 중 애플리케이션 동작을 테스트합니다.
- 확장 문제는 초기에 알리기 - 리샤딩은 컴퓨팅 집약적인 작업입니다. 이 때문에 리샤딩 중에는 CPU 사용률을 멀티 코어 인스턴스의 경우, 80% 미만으로, 싱글 코어 인스턴스의 경우에는 50% 미만으로 유지하는 것이 좋습니다. 애플리케이션에서 규모 조정 문제 관찰을 시작하기 전에 MemoryDB 측정치를 모니터링한 다음 리샤딩을 시작합니다. `CPUUtilization`, `NetworkBytesIn`, `NetworkBytesOut`, `CurrConnections`, `NewConnections`, `FreeableMemory`, `SwapUsage` 및 `BytesUsedForMemoryDB` 지표를 추적하면 유용합니다.
- 스케일 인 전 충분한 여유 메모리를 사용할 수 있는지 확인 - 확장하는 경우, 샤드에서 제거하려는 사용 중인 메모리의 1.5배가 넘는 여유 메모리가 샤드에 있는지 확인합니다.
- 사용량이 적은 시간에 리샤딩 시작 - 이 사례를 적용하면 리샤딩 작업 중 클라이언트에서 지연 시간과 처리량에 미치는 영향을 줄일 수 있습니다. 또한 슬롯 재분산에 더 많은 리소스를 사용할 수 있기 때문에 리샤딩을 더욱 빠르게 완료할 수 있습니다.
- 클라이언트 제한 시간 동작 검토 - 일부 클라이언트에서는 온라인 클러스터 크기 조정 중 지연 시간이 더 길어지는 경우를 관찰할 수 있습니다. 제한 시간을 좀 더 길게 설정해 클라이언트 라이브러리를 구성하면 서버에 대한 로드가 더욱 큰 상황에서도 시스템에서 연결할 시간을 가질 수 있습니다. 일부 경우에 서버에 대해 많은 수의 연결을 열 수 있습니다. 이 경우 지수 백오프를 추가해 로직을 다시 연결합니다. 이렇게 하면 동시에 서버에 연결하는 새로운 연결이 급격하게 증가하는 것을 방지할 수 있습니다.

리샤딩 중에 다음 작업을 수행하는 것이 좋습니다.

- 비용이 많이 드는 명령 피하기 - 컴퓨팅 및 I/O 집약적인 작업(예: `KEYS` 및 `SMEMBERS` 명령)을 실행하지 않습니다. 이러한 작업은 클러스터에 대한 부하를 늘리고 클러스터의 성능에 영향을 미치기 때문에 이러한 접근 방식을 채택하는 것이 좋습니다. 대신 `SCAN` 및 `SSCAN` 명령을 사용합니다.
- Lua 모범 사례 따르기 - 오래 실행되는 Lua 스크립트의 사용을 피하고 항상 사전에 Lua 스크립트에 사용되는 키를 선언합니다. 이렇게 하면 Lua 스크립트가 슬롯 간 명령을 사용하지 않습니다. Lua 스크립트에 사용되는 키가 동일한 슬롯에 속해 있는지 확인하세요.

리샤딩 후에는 다음 내용에 유념하세요.

- 대상 샤드에 사용 가능한 메모리가 부족한 경우에는 부분적인 확장만 가능했을 수 있습니다. 이러한 결과가 발생하면 필요한 경우, 사용 가능한 메모리를 검토한 후 작업을 다시 시도하세요.
- 항목이 많은 슬롯은 마이그레이션되지 않습니다. 특히, 직렬화 후 크기가 256MB 이상인 항목이 있는 슬롯은 마이그레이션되지 않습니다.
- 리샤딩 작업 중에는 Lua 스크립트 내에서 `FLUSHALL`과 `FLUSHDB` 명령이 지원되지 않습니다.

MemoryDB 복제 이해

MemoryDB는 최대 500개 샤드에 분할된 데이터로 복제를 구현합니다.

클러스터의 각 샤드에는 읽기/쓰기 프라이머리 노드 하나와 최대 5개의 읽기 전용 복제본 노드가 있습니다. 각 프라이머리 노드는 최대 100MB/s까지 지원할 수 있습니다. 하나의 클러스터당 최대 500개의 노드로 구성된 더 많은 수의 샤드와 더 적은 수의 복제본을 가진 클러스터를 생성할 수 있습니다. 이 클러스터 구성은 500개의 샤드 및 0개의 복제본부터 100개의 샤드 및 4개의 복제본까지 해당될 수 있으며, 이는 허용되는 최대 복제본 수입니다.

일관성

MemoryDB에서 프라이머리 노드는 매우 일관적입니다. 성공적인 쓰기 작업은 클라이언트로 돌아가기 전에 분산된 다중 AZ 트랜잭션 로그에 안정적으로 저장됩니다. 프라이머리에서의 읽기 작업은 항상 이전에 성공한 모든 쓰기 작업의 영향을 반영하는 최신 데이터를 반환합니다. 이러한 강력한 일관성은 기본 장애 조치 전반에서 유지됩니다.

MemoryDB의 복제본 노드는 결국에 일관되게 이루어집니다. CloudWatch에 지연 지표가 게시된 경우, 복제본의 읽기 작업(`READONLY` 명령 사용) 이 가장 최근의 성공적인 쓰기 작업의 영향을 반영하지 않을 수 있습니다. 하지만 단일 복제본의 읽기 작업은 순차적으로 일관됩니다. 성공적인 쓰기 작업은 프라이머리에서 실행된 순서와 동일한 순서로 각 복제본에 적용됩니다.

클러스터에서의 복제

샤드의 각 읽기 복제본은 샤드의 프라이머리 노드에서 가져온 데이터 사본을 유지합니다. 트랜잭션 로그를 사용하는 비동기식 복제 메커니즘은 읽기 복제본이 기본 노드와 동기화되어 있는 상태를 유지하는 데 사용됩니다. 애플리케이션은 클러스터의 모든 노드로부터 읽을 수 있습니다. 애플리케이션은 기본 노드에만 쓸 수 있습니다. 읽기 복제본은 읽기 확장성을 개선합니다. MemoryDB는 데이터를 내구성이 뛰어난 트랜잭션 로그에 저장하므로 데이터가 손실될 위험이 없습니다. 데이터는 MemoryDB 클러스터의 샤드 간에 파티셔닝됩니다.

애플리케이션은 MemoryDB 클러스터의 클러스터 엔드포인트를 사용하여 클러스터의 노드와 연결합니다. 자세한 내용은 연결 엔드포인트 찾기 섹션을 참조하세요.

MemoryDB 클러스터는 지역적이며 한 리전의 노드만 포함할 수 있습니다. 내결함성을 개선하기 위해 해당 리전 내의 여러 가용 영역에 걸쳐 프라이머리와 읽기 복제본을 모두 프로비저닝해야 합니다.

모든 MemoryDB 클러스터에는 다중 AZ를 제공하는 복제를 사용하는 것이 좋습니다. 자세한 내용은 다중 AZ로 MemoryDB의 가동 중지 시간 최소화 섹션을 참조하세요.

다중 AZ로 MemoryDB의 가동 중지 시간 최소화

MemoryDB가 프라이머리 노드를 대체해야 하는 여러 가지 경우가 있습니다. 이러한 경우에는 특정 유형의 계획적인 유지 관리 및 드물지만 프라이머리 노드나 가용 영역에 장애가 발생하는 경우가 포함됩니다.

노드 장애에 대한 대응은 장애가 발생한 노드에 따라 달라집니다. 그러나 모든 경우에 MemoryDB는 노드 교체 또는 장애 조치 중에 데이터가 손실되지 않도록 합니다. 예를 들어, 복제본에 장애가 발생하면 장애가 발생한 노드가 교체되고 트랜잭션 로그에서 데이터가 동기화됩니다. 프라이머리 노드에 장애가 발생하면 장애 조치 중에 데이터가 손실되지 않도록 일관된 복제본으로 장애 조치가 트리거됩니다. 이제 새 프라이머리 노드에서 쓰기가 제공됩니다. 그러면 이전 프라이머리 노드가 교체되고 트랜잭션 로그와 동기화됩니다.

단일 노드 샤드(복제본 없음)에서 프라이머리 노드에 장애가 발생하면 프라이머리 노드가 교체되어 트랜잭션 로그와 동기화될 때까지 MemoryDB는 쓰기 수락을 중단합니다.

노드 교체로 인해 클러스터에 약간의 가동 중지가 발생할 수 있지만, 다중 AZ를 활성화된 경우 가동 중지 시간이 최소화됩니다. 프라이머리 노드의 역할은 자동으로 복제본 중 하나로 장애 조치됩니다. MemoryDB가 이 장애 조치를 투명하게 처리하기 때문에 새로운 프라이머리 노드를 생성하고 프로비저닝할 필요가 없습니다. 이 장애 조치 및 복제본 승격을 통해 승격이 완료되는 즉시 새 기본 노드에 작성을 재개할 수 있습니다.

유지 관리 업데이트 또는 서비스 업데이트로 인해 시작된 계획된 노드 교체의 경우, 클러스터가 들어오는 쓰기 요청을 처리하는 동안 계획된 노드 교체가 완료된다는 점에 유의하세요.

MemoryDB 클러스터의 다중 AZ를 사용하면 내결함성이 향상됩니다. 특히 클러스터의 프라이머리 노드에 접속할 수 없거나 어떤 이유로든 실패하는 경우에 그렇습니다. MemoryDB 클러스터의 다중 AZ는 각 샤드에 두 개 이상의 노드가 있어야 하며 자동으로 활성화됩니다.

주제

- [다중 AZ 응답이 있는 장애 시나리오](#)
- [자동 장애 조치 테스트](#)

다중 AZ 응답이 있는 장애 시나리오

다중 AZ가 활성화된 경우, 장애가 발생한 프라이머리 노드는 사용 가능한 복제본으로 장애 조치됩니다. 복제본은 트랜잭션 로그와 자동으로 동기화되고 프라이머리 노드가 되므로 프라이머리 노드를 새로 만들고 다시 프로비저닝하는 것보다 훨씬 빠릅니다. 이 프로세스는 보통 클러스터에 다시 작성하려면 몇 초 정도 소요됩니다.

다중 AZ가 활성화된 경우, MemoryDB가 프라이머리 노드의 상태를 지속적으로 모니터링합니다. 기본 노드에 장애가 발생하면 장애 유형에 따라 다음 작업 중 하나가 수행됩니다.

주제

프라이머리 노드에만 장애가 발생한 경우의 장애 시나리오

프라이머리 노드에만 장애가 발생한 경우, 복제본은 자동으로 프라이머리 노드가 됩니다. 그러면 대체 복제본이 생성되어 장애가 발생한 프라이머리 노드와 동일한 가용 영역에 프로비저닝됩니다.

프라이머리 노드에만 장애가 발생한 경우, MemoryDB 다중 AZ는 다음 작업을 수행합니다.

1. 장애가 발생한 기본 노드는 오프라인 상태로 전환됩니다.
2. 최신 복제본은 자동으로 기본 복제본이 됩니다.

   장애 조치 프로세스가 완료되는 즉시 쓰기를 재개할 수 있으며 일반적으로 몇 초 정도 소요됩니다.
3. 대체 복제본을 시작하고 프로비저닝합니다.

   장애가 발생한 프라이머리 노드가 있는 가용 영역에서 대체 복제본을 시작하여 노드 배포를 유지합니다.
4. 복제본은 트랜잭션 로그와 동기화됩니다.

클러스터의 엔드포인트를 찾는 방법에 대한 정보는 다음 항목을 참조하세요.

- MemoryDB 클러스터의 엔드포인트 찾기(MemoryDB API)

프라이머리 노드 및 일부 복제본에 장애가 발생한 경우의 장애 시나리오

기본 복제본과 하나 이상의 복제본에 오류가 발생하면 최신 복제본이 기본 클러스터로 승격됩니다. 장애가 발생한 노드와 동일 가용 영역에 새로운 복제본이 생성되고 프로비저닝됩니다.

프라이머리 노드와 일부 복제본에 장애가 발생한 경우, MemoryDB 다중 AZ는 다음 작업을 수행합니다.

1. 장애가 발생한 프라이머리 노드 및 장애가 발생한 복제본이 오프라인 상태로 전환됩니다.
2. 사용 가능한 복제본이 프라이머리 노드가 됩니다.

   장애 조치가 완료되는 즉시 쓰기를 재개할 수 있으며 일반적으로 몇 초 정도 소요됩니다.
3. 교체용 복제본을 생성하고 프로비저닝합니다.

   장애가 발생한 노드의 가용 영역에서 교체용 복제본을 생성하여 노드 배포를 유지합니다.
4. 모든 노드가 트랜잭션 로그와 동기화됩니다.

클러스터의 엔드포인트를 찾는 방법에 대한 정보는 다음 항목을 참조하세요.

- MemoryDB 클러스터(CLI AWS)의 엔드포인트 찾기
- MemoryDB 클러스터의 엔드포인트 찾기(MemoryDB API)

전체 클러스터에 장애가 발생한 경우의 장애 시나리오

모든 것에 장애가 발생하면 모든 노드를 동일한 가용 영역에 원본 노드로 재생성하고 프로비저닝합니다.

이 시나리오에서는 데이터가 트랜잭션 로그에 유지되었으므로 데이터 손실은 없습니다.

전체 클러스터에 장애가 발생한 경우, MemoryDB 다중 AZ는 다음 작업을 수행합니다.

1. 장애가 발생한 프라이머리 노드 및 복제본이 오프라인 상태로 전환됩니다.
2. 대체 프라이머리 노드가 생성되고 프로비저닝되며 트랜잭션 로그와 동기화됩니다.
3. 대체 복제본이 생성되고 프로비저닝되며 트랜잭션 로그와 동기화됩니다.

   장애가 발생한 노드의 가용 영역에서 대체를 생성하여 노드 배포를 유지합니다.

클러스터의 엔드포인트를 찾는 방법에 대한 정보는 다음 항목을 참조하세요.

- MemoryDB 클러스터(CLI AWS)의 엔드포인트 찾기
- MemoryDB 클러스터의 엔드포인트 찾기(MemoryDB API)

자동 장애 조치 테스트

MemoryDB 콘솔, 및 AWS CLI, MemoryDB API를 사용하여 자동 장애 조치를 테스트할 수 있습니다.

테스트 시 다음 사항에 유의하세요.

- 24시간 동안 이 작업을 최대 5회까지 사용할 수 있습니다.
- 다른 클러스터에 있는 샤드에서 이 작업을 직접 호출하는 경우, 동시에 직접 호출할 수 있습니다.
- 경우에 따라 동일한 MemoryDB 클러스터의 서로 다른 샤드에서 이 작업을 여러 번 호출할 수 있습니다. 이러한 경우 후속 호출이 이루어지기 전에 첫 번째 노드 교체가 완료되어야 합니다.
- 노드 교체가 완료되었는지 확인하려면 MemoryDB 콘솔, AWS CLI 또는 MemoryDB API를 사용하여 이벤트를 점검합니다. 발생 순서대로 나열되어 있는 아래 목록에서 다음과 같은 `FailoverShard` 관련 이벤트를 찾습니다.
  1. 클러스터 메시지: `FailoverShard API called for shard <shard-id>`
  2. 클러스터 메시지: `Failover from primary node <primary-node-id> to replica node <node-id> completed`
  3. 클러스터 메시지: `Recovering nodes <node-id>`
  4. 클러스터 메시지: `Finished recovery for nodes <node-id>`

  자세한 내용은 다음 자료를 참조하세요.

  - MemoryDB API 참조의 [DescribeEvents]
- 이 API는 MemoryDB 장애 조치의 경우, 애플리케이션의 동작을 테스트하도록 설계되었습니다. 클러스터 문제를 해결하기 위해 장애 조치를 시작하는 운영 도구로 설계되지 않았습니다. 또한 대규모 운영 이벤트와 같은 특정 조건에서는 AWS가 이 API를 차단할 수 있습니다.

주제

- [AWS Management 콘솔을 사용하여 자동 장애 조치 테스트]
- [AWS CLI를 사용하여 자동 장애 조치 테스트]
- [MemoryDB API를 사용하여 자동 장애 조치 테스트]

AWS Management 콘솔을 사용하여 자동 장애 조치 테스트

다음 절차에 따라 콘솔로 자동 장애 조치를 테스트합니다.

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 테스트할 클러스터 왼쪽에 있는 라디오 버튼을 선택합니다. 이 클러스터에는 복제본 노드가 하나 이상 있어야 합니다.
3. [Details] 영역에서 이 클러스터가 다중 AZ 활성 상태인지 확인합니다. 해당 클러스터가 다중 AZ 활성 상태가 아닌 경우 다른 클러스터를 선택하거나 다중 AZ를 활성화하도록 이 클러스터를 수정합니다. 자세한 내용은 MemoryDB 클러스터 수정 섹션을 참조하세요.
4. 클러스터의 이름을 선택합니다.
5. 샤드 및 노드 페이지에서 장애 조치를 테스트할 샤드에 대해 샤드 이름을 선택합니다.
6. 노드는 [Failover Primary]를 선택합니다.
7. 기본 노드를 장애 조치하려면 [Continue]를 선택하고 작업을 취소하여 기본 노드를 장애 조치하지 않으려면 [Cancel]을 선택합니다.

   장애 조치 프로세스 중에 콘솔은 노드 상태를 계속해서 사용 가능으로 표시합니다. 장애 조치 테스트 진행률을 추적하려면 콘솔 탐색 창에서 [Events]를 선택합니다. [Events] 탭에서 장애 조치의 시작(`FailoverShard API called`) 및 완료(`Recovery completed`)를 나타내는 이벤트를 주시합니다.

AWS CLI를 사용하여 자동 장애 조치 테스트

AWS CLI 작업 failover-shard를 사용하여 모든 다중 AZ가 활성화된 클러스터에서 자동 장애 조치를 테스트할 수 있습니다.

파라미터

- `--cluster-name` - 필수입니다. 테스트할 클러스터입니다.
- `--shard-name` - 필수입니다. 자동 장애 조치를 테스트할 샤드의 이름입니다. 24시간 동안 최대 5개의 샤드를 테스트할 수 있습니다.

다음 예제에서는 AWS CLI을(를) 사용하여 MemoryDB 클러스터 `my-cluster`의 샤드 `0001`에서 `failover-shard`을(를) 호출합니다.

Linux, macOS, Unix의 경우:

```
aws memorydb failover-shard \
   --cluster-name my-cluster \
   --shard-name 0001
```

Windows의 경우:

```
aws memorydb failover-shard ^
   --cluster-name my-cluster ^
   --shard-name 0001
```

장애 조치 진행률을 추적하려면 AWS CLI `describe-events` 작업을 사용하세요.

다음과 같은 JSON 응답을 반환합니다.

```
{
    "Events": [
        {
            "SourceName": "my-cluster",
            "SourceType": "cluster",
            "Message": "Failover to replica node my-cluster-0001-002 completed",
            "Date": "2021-08-22T12:39:37.568000-07:00"
        },
        {
            "SourceName": "my-cluster",
            "SourceType": "cluster",
            "Message": "Starting failover for shard 0001",
            "Date": "2021-08-22T12:39:10.173000-07:00"
        }
    ]
}
```

자세한 내용은 다음 자료를 참조하세요.

- failover-shard
- describe-events

MemoryDB API를 사용하여 자동 장애 조치 테스트

다음 예시는 클러스터 `memorydb00`의 샤드 `0003`에서 `FailoverShard`을(를) 직접적으로 호출합니다.

Example 자동 장애 조치 테스트

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=FailoverShard
    &ShardName=0003
    &ClusterName=memorydb00
    &Version=2021-01-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210801T192317Z
    &X-Amz-Credential=<credential>
```

장애 조치 진행률을 추적하려면 MemoryDB `DescribeEvents` API 작업을 사용하세요.

자세한 내용은 다음 자료를 참조하세요.

- [FailoverShard]
- [DescribeEvents]

복제본 수 변경

AWS Management 콘솔, AWS CLI 또는 MemoryDB API를 사용해 MemoryDB 클러스터의 읽기 복제본 수를 동적으로 늘리거나 줄일 수 있습니다. 모든 샤드의 복제본 수는 같아야 합니다.

클러스터의 복제본 수 늘리기

MemoryDB 클러스터의 복제본 수를 샤드당 최대 5개까지 늘릴 수 있습니다. AWS Management 콘솔, AWS CLI, 또는 MemoryDB API를 사용해 늘릴 수 있습니다.

주제

- AWS Management 콘솔 사용
- AWS CLI 사용
- MemoryDB API 사용

AWS Management 콘솔 사용

MemoryDB 클러스터(콘솔)의 복제본 수를 늘리려면 클러스터에서 노드 추가/제거을(를) 참조하세요.

AWS CLI 사용

MemoryDB 클러스터의 복제본 수를 늘리려면 다음 파라미터와 함께 `update-cluster` 명령을 사용합니다.

- `--cluster-name` - 필수입니다. 복제본 수를 늘리려는 클러스터를 식별합니다.
- `--replica-configuration` - 필수입니다. 복제본 수를 설정할 수 있습니다. 복제본 수를 늘리려면 이 작업이 끝날 때 `ReplicaCount` 속성을 이 샤드의 원하는 복제본 수로 설정합니다.

Example

다음은 클러스터 `my-cluster`의 복제본 수를 2로 늘리는 예입니다.

Linux, macOS, Unix의 경우:

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --replica-configuration \
        ReplicaCount=2
```

Windows의 경우:

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --replica-configuration ^
```

```
        ReplicaCount=2
```

다음과 같은 JSON 응답을 반환합니다.

```
{
    "Cluster": {
        "Name": "my-cluster",
        "Status": "updating",
        "NumberOfShards": 1,
        "ClusterEndpoint": {
            "Address": "clustercfg.my-cluster.xxxxx.memorydb.us-east-1.amazonaws.com",
            "Port": 6379
        },
        "NodeType": "db.r6g.large",
        "EngineVersion": "6.2",
        "EnginePatchVersion": "6.2.6",
        "ParameterGroupName": "default.memorydb-redis6",
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
        "SnapshotWindow": "04:30-05:30",
        "DataTiering": "false",
        "AutoMinorVersionUpgrade": true
    }
}
```

상태가 업데이트 중에서 사용 가능으로 변경된 후 업데이트된 클러스터의 세부 정보를 보려면 다음 명령을 사용하세요.

Linux, macOS, Unix의 경우:

```
aws memorydb describe-clusters \
    --cluster-name my-cluster
    --show-shard-details
```

Windows의 경우:

```
aws memorydb describe-clusters ^
    --cluster-name my-cluster
```

```
    --show-shard-details
```

다음과 같은 JSON 응답을 반환합니다.


```
{
    "Clusters": [
        {
            "Name": "my-cluster",
            "Status": "available",
            "NumberOfShards": 1,
            "Shards": [
                {
                    "Name": "0001",
                    "Status": "available",
                    "Slots": "0-16383",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0001-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0001-002",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1b",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0001-003",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
```

```
                            "CreateTime": "2021-08-22T12:59:31.844000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        }
                    ],
                    "NumberOfNodes": 3
                }
            ],
            "ClusterEndpoint": {
                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
            },
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "my-sg",
            "TLSEnabled": true,
            "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
            "MaintenanceWindow": "wed:03:00-wed:04:00",
            "SnapshotWindow": "04:30-05:30",
            "ACLName": "my-acl",
            "DataTiering": "false",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```

CLI를 사용하여 복제본 수를 늘리는 방법에 대한 자세한 내용은 AWS CLI 명령 참조의 [update-cluster](#) 항목을 참조하세요.

MemoryDB API 사용

MemoryDB 샤드의 복제본 수를 늘리려면 다음 파라미터와 함께 `UpdateCluster` 작업을 사용합니다.

- `ClusterName` - 필수입니다. 복제본 수를 늘리려는 클러스터를 식별합니다.

- `ReplicaConfiguration` - 필수입니다. 복제본 수를 설정할 수 있습니다. 복제본 수를 늘리려면 이 작업이 끝날 때 `ReplicaCount` 속성을 이 샤드에 포함할 복제본 수로 설정하세요.

Example

다음은 클러스터 `sample-cluster`의 복제본 수를 3으로 늘리는 예입니다. 예제가 완료되면 각 샤드에 복제본 3개가 있습니다. 이 숫자는 단일 샤드가 있는 MemoryDB 클러스터이든 여러 샤드가 있는 MemoryDB 클러스터이든 상관없이 적용됩니다.

```
https://memory-db.us-east-1.amazonaws.com/
      ?Action=UpdateCluster
      &ReplicaConfiguration.ReplicaCount=3
      &ClusterName=sample-cluster
      &Version=2021-01-01
      &SignatureVersion=4
      &SignatureMethod=HmacSHA256
      &Timestamp=20210802T192317Z
      &X-Amz-Credential=<credential>
```

API를 사용하여 복제본 수를 늘리는 것에 대한 자세한 내용은 UpdateCluster를 참조하세요.

클러스터의 복제본 수 줄이기

MemoryDB용 클러스터의 복제본 수를 줄일 수 있습니다. 복제본 수를 0으로 줄일 수 있지만 프라이머리 노드에 장애가 발생하면 복제본으로 장애 조치할 수 없습니다.

AWS Management 콘솔, AWS CLI 또는 MemoryDB API를 사용하여 클러스터의 복제본 수를 줄일 수 있습니다.

주제

AWS Management 콘솔 사용

MemoryDB 클러스터(콘솔)의 복제본 수를 줄이려면 클러스터에서 노드 추가/제거을(를) 참조하세요.

AWS CLI 사용

MemoryDB 클러스터의 복제본 수를 줄이려면 다음 파라미터와 함께 `update-cluster` 명령을 사용합니다.

- `--cluster-name` - 필수입니다. 복제본 수를 줄이려는 클러스터를 식별합니다.
- `--replica-configuration` - 필수입니다.

  `ReplicaCount`- 이 속성을 설정하여 원하는 복제본 노드의 수를 지정합니다.

Example

다음은 `--replica-configuration`을 사용해 클러스터 `my-cluster`의 복제본 수를 지정된 값으로 줄이는 예입니다.

Linux, macOS, Unix의 경우:

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --replica-configuration \
        ReplicaCount=1
```

Windows의 경우:

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --replica-configuration ^
        ReplicaCount=1 ^
```

다음과 같은 JSON 응답을 반환합니다.


```
{
    "Cluster": {
        "Name": "my-cluster",
        "Status": "updating",
        "NumberOfShards": 1,
        "ClusterEndpoint": {
            "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-east-1.amazonaws.com",
            "Port": 6379
        },
        "NodeType": "db.r6g.large",
        "EngineVersion": "6.2",
        "EnginePatchVersion": "6.2.6",
        "ParameterGroupName": "default.memorydb-redis6",
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
        "SnapshotWindow": "04:30-05:30",
        "DataTiering": "false",
        "AutoMinorVersionUpgrade": true
    }
}
```


업데이트 중에서 사용 가능으로 변경된 후 업데이트된 클러스터의 세부 정보를 보려면 다음 명령을 사용하세요.

Linux, macOS, Unix의 경우:

```
aws memorydb describe-clusters \
    --cluster-name my-cluster
    --show-shard-details
```

Windows의 경우:

```
aws memorydb describe-clusters ^
    --cluster-name my-cluster
    --show-shard-details
```

다음과 같은 JSON 응답을 반환합니다.

```
{
    "Clusters": [
        {
            "Name": "my-cluster",
            "Status": "available",
            "NumberOfShards": 1,
            "Shards": [
                {
                    "Name": "0001",
                    "Status": "available",
                    "Slots": "0-16383",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0001-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0001-002",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1b",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        }
                    ],
                    "NumberOfNodes": 2
```

```
                }
            ],
            "ClusterEndpoint": {
                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
            },
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "my-sg",
            "TLSEnabled": true,
            "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
            "MaintenanceWindow": "wed:03:00-wed:04:00",
            "SnapshotWindow": "04:30-05:30",
            "ACLName": "my-acl",
            "DataTiering": "false",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```

CLI를 사용하여 복제본 수를 줄이는 방법에 대한 자세한 내용은 AWS CLI 명령 참조의 update-cluster 항목을 참조하세요.

MemoryDB API 사용

MemoryDB 클러스터의 복제본 수를 줄이려면 다음 파라미터와 함께 `UpdateCluster` 작업을 사용합니다.

- `ClusterName` - 필수입니다. 복제본 수를 줄이려는 클러스터를 식별합니다.
- `ReplicaConfiguration` - 필수입니다. 복제본 수를 설정할 수 있습니다.

  `ReplicaCount`- 이 속성을 설정하여 원하는 복제본 노드의 수를 지정합니다.

Example

다음은 `ReplicaCount`를 사용해 클러스터 `sample-cluster`의 복제본 수를 1로 줄이는 예입니다. 예제가 완료되면 각 샤드에 복제본 1개가 있습니다. 이 숫자는 단일 샤드가 있는 MemoryDB 클러스터이든 여러 샤드가 있는 MemoryDB 클러스터이든 상관없이 적용됩니다.

```
https://memory-db.us-east-1.amazonaws.com/
      ?Action=UpdateCluster
      &ReplicaConfiguration.ReplicaCount=1
      &ClusterName=sample-cluster
      &Version=2021-01-01
      &SignatureVersion=4
      &SignatureMethod=HmacSHA256
      &Timestamp=20210802T192317Z
      &X-Amz-Credential=<credential>
```

API를 사용하여 복제본 수를 줄이는 것에 대한 자세한 내용은 UpdateCluster를 참조하세요.

스냅샷 및 복원

MemoryDB 클러스터는 자동으로 데이터를 다중 AZ 트랜잭션 로그에 백업하지만, 클러스터의 특정 시점 스냅샷을 주기적으로 또는 필요에 따라 생성하도록 선택할 수 있습니다. 이러한 스냅샷을 사용하여 이전 시점에서 클러스터를 다시 만들거나 새 클러스터를 시드할 수 있습니다. 스냅샷은 클러스터의 모든 데이터와 클러스터의 메타데이터로 구성됩니다. 모든 스냅샷은 Amazon Simple Storage Service(S3)에 쓰여지므로 내구성 있는 스토리지가 확보됩니다. 언제든지 새로운 MemoryDB 클러스터를 생성하고 스냅샷의 데이터로 클러스터를 채워 데이터를 복원할 수 있습니다. MemoryDB를 사용하면 , AWS Command Line Interface(AWS CLI) 및 MemoryDB APIAWS Management 콘솔를 사용하여 스냅샷을 관리할 수 있습니다.

주제

- 스냅샷 내보내기
- 스냅샷에서 복원
- 외부에서 생성된 스냅샷으로 새 클러스터 시드
- 스냅샷 태깅
- 스냅샷 삭제

스냅샷 제약 조건

스냅샷을 계획하거나 만들려는 경우, 다음 제약 조건을 고려해야 합니다.

- MemoryDB 클러스터의 경우, 지원되는 모든 노드 유형에서 스냅샷 및 복원을 사용할 수 있습니다.
- 24시간 연속으로 클러스터당 20개 이내의 수동 스냅샷을 만들 수 있습니다.
- MemoryDB는 클러스터 수준에서의 스냅샷 촬영만 지원합니다. MemoryDB는 샤드 또는 노드 수준에서 스냅샷을 찍는 것을 지원하지 않습니다.
- 스냅샷 프로세스 중에는 클러스터에서 다른 API 또는 CLI 작업을 실행할 수 없습니다.
- 클러스터를 삭제하고 최종 스냅샷을 요청할 경우, MemoryDB는 언제나 프라이머리 노드에서 스냅샷을 만듭니다. 그러면 클러스터가 삭제되기 전에 가장 최신 데이터를 캡처할 수 있습니다.

스냅샷 비용

MemoryDB를 사용하여 활성 MemoryDB 클러스터마다 스냅샷 하나를 무료로 저장할 수 있습니다. 추가 스냅샷을 위한 스토리지 공간은 모든 AWS 리전에 대해 매월 $0.085/GB의 요금이 청구됩니다. 스냅샷을 만들거나 스냅샷의 데이터를 MemoryDB 클러스터에 복원하는 경우에는 데이터 전송 요금이 없습니다.

자동 스냅샷 예약

모든 MemoryDB 클러스터에서 자동 스냅샷을 활성화할 수 있습니다. 자동 스냅샷을 활성화할 경우, MemoryDB에서는 매일 클러스터 스냅샷을 생성합니다. 클러스터에는 영향을 주지 않으며 변경 사항이 즉시 적용됩니다. 자세한 내용은 스냅샷에서 복원 단원을 참조하십시오.

자동 스냅샷을 예약할 때는 다음 설정을 계획해야 합니다.

- 스냅샷 기간 - MemoryDB에서 스냅샷을 만들기 시작하는 하루 중 기간입니다. 스냅샷 기간의 최소 길이는 60분입니다. 아무 때나 편리한 시간 또는 하루 중 스냅샷을 피할 시간, 특히 사용량이 많은 기간의 스냅샷 기간을 설정할 수 있습니다.

  스냅샷 기간을 지정하지 않으면 MemoryDB에서 자동으로 할당합니다.
- 스냅샷 보존 제한 - Amazon S3에서 스냅샷이 보존되는 일수입니다. 예를 들어, 보존 제한을 5로 설정하면 오늘 만든 스냅샷이 5일간 보존됩니다. 보존 제한이 만료되면 스냅샷이 자동으로 삭제됩니다.

  최대 스냅샷 보존 제한은 35일입니다. 스냅샷 보존 제한을 0으로 설정하면 클러스터의 자동 스냅샷이 비활성화됩니다. 자동 스냅샷 기능이 비활성화된 상태에서도 MemoryDB 데이터는 여전히 완전한 내구성을 유지합니다.

MemoryDB 콘솔AWS CLI, 또는 MemoryDB API를 사용하여 MemoryDB 클러스터를 생성할 때 자동 스냅샷을 활성화하거나 비활성화할 수 있습니다. MemoryDB 클러스터를 생성할 때 스냅샷 섹션에서 자동 스냅샷 활성화 확인란을 선택하여 자동 스냅샷을 활성화할 수 있습니다. 자세한 설명은 MemoryDB 클러스터 생성 섹션을 참조하세요.

수동 스냅샷 생성

자동 스냅샷 외에도 언제든지 수동으로 스냅샷을 만들 수 있습니다. 지정한 보존 기간 후에 자동으로 삭제되는 자동 스냅샷과 달리 수동 스냅샷에는 나중에 자동으로 삭제되는 보존 기간이 없습니다. 수동 스냅샷은 수동으로 삭제해야 합니다. 클러스터나 노드를 삭제하더라도 해당 클러스터나 노드의 수동 스냅샷은 보존됩니다. 수동 스냅샷을 더 이상 보존하지 않으려면 이 스냅샷을 직접 명시적으로 삭제해야 합니다.

수동 스냅샷은 테스트와 아카이빙에 유용합니다. 예를 들어, 테스트 목적으로 기준 데이터 세트를 개발했다고 가정해 보겠습니다. 데이터의 수동 스냅샷을 만들고 언제든지 복원할 수 있습니다. 데이터를 수정하는 애플리케이션을 테스트한 후 새로운 클러스터를 만들고 기준 스냅샷에서 복원하여 데이터를 재설정할 수 있습니다. 클러스터가 준비되면 기준 데이터와 비교하여 애플리케이션을 다시 테스트하고 필요한 만큼 이 프로세스를 반복할 수 있습니다.

수동 스냅샷을 직접 생성할 뿐 아니라 다음 방법 중 하나로 수동 스냅샷을 생성할 수 있습니다.

- 스냅샷 복사 - 소스 스냅샷을 자동으로 생성했는지 수동으로 생성했는지는 중요하지 않습니다.
- 최종 스냅샷 생성 - 클러스터를 삭제하기 직전에 스냅샷을 생성합니다.

기타 중요한 주제

- 스냅샷 제약 조건
- 스냅샷 비용

AWS CLI, 또는 MemoryDB API를 사용하여 노드AWS Management 콘솔의 수동 스냅샷을 생성할 수 있습니다.

수동 스냅샷 생성(콘솔)

클러스터(콘솔)의 스냅샷을 생성하려면

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/memorydb/ MemoryDB 콘솔을 엽니다.
2. 좌측 탐색 창에서 클러스터(Clusters)를 선택합니다.

   MemoryDB 클러스터 화면이 나타납니다.
3. 백업할 MemoryDB 클러스터 이름 왼쪽의 라디오 버튼을 선택합니다.

4. 작업을 선택한 다음 스냅샷 만들기를 선택합니다.
5. 스냅샷 창의 스냅샷 이름 상자에 스냅샷 이름을 입력합니다. 이름은 스냅샷된 클러스터와 스냅샷 날짜 및 시간을 나타내는 것이 좋습니다.

   클러스터 명명 제약 조건은 다음과 같습니다.

   - 1~40자의 영숫자 또는 하이픈으로 구성되어야 합니다.
   - 문자로 시작해야 합니다.
   - 하이픈 2개가 연속될 수 없습니다.
   - 끝에 하이픈이 올 수 없습니다.
6. 암호화 옵션에서 기본 암호화 키를 사용할지 또는 고객 관리 키를 사용할지 선택합니다. 자세한 내용은 MemoryDB에서 전송 중 데이터 암호화(TLS) 단원을 참조하십시오.
7. 태그에서 선택적으로 태그를 추가하여 스냅샷을 검색 및 필터링하거나 AWS비용을 추적합니다.
8. 스냅샷 생성(Take Snapshot)을 선택합니다.

   클러스터 상태가 snapshotting으로 바뀝니다. 상태가 다시 available로 바뀌면 스냅샷이 완료됩니다.

수동 스냅샷 생성(AWSCLI)

를 사용하여 클러스터의 수동 스냅샷을 생성하려면 다음 파라미터와 함께 `create-snapshot`AWS CLI작업을 AWS CLI사용합니다.

- `--cluster-name` - 스냅샷의 소스로 사용할 MemoryDB 클러스터의 이름입니다. MemoryDB 클러스터를 백업할 때에만 이 파라미터를 사용하세요.

  클러스터 명명 제약 조건은 다음과 같습니다.

  - 1~40자의 영숫자 또는 하이픈으로 구성되어야 합니다.
  - 문자로 시작해야 합니다.
  - 하이픈 2개가 연속될 수 없습니다.
  - 끝에 하이픈이 올 수 없습니다.

- `--snapshot-name` - 생성할 스냅샷의 이름입니다.

관련 주제

자세한 내용은 AWS CLI 명령 참조의 `create-snapshot` 섹션을 참조하십시오.

수동 스냅샷 생성(MemoryDB API)

MemoryDB API를 사용하여 클러스터의 수동 스냅샷을 생성하려면 다음 파라미터와 함께 `CreateSnapshot` MemoryDB API 작업을 사용하세요.

- `ClusterName` - 스냅샷의 소스로 사용할 MemoryDB 클러스터의 이름입니다. MemoryDB 클러스터를 백업할 때에만 이 파라미터를 사용하세요.

  클러스터 명명 제약 조건은 다음과 같습니다.
  - 1~40자의 영숫자 또는 하이픈으로 구성되어야 합니다.
  - 문자로 시작해야 합니다.
  - 하이픈 2개가 연속될 수 없습니다.
  - 끝에 하이픈이 올 수 없습니다.
- `SnapshotName` - 생성할 스냅샷의 이름입니다.

관련 주제

자세한 내용은 CreateSnapshot를 참조하세요.

최종 스냅샷 생성

MemoryDB 콘솔AWS CLI, 또는 MemoryDB API를 사용하여 최종 스냅샷을 생성할 수 있습니다.

최종 스냅샷 생성 (콘솔)

MemoryDB 콘솔을 사용하여 MemoryDB 클러스터를 삭제할 때 최종 스냅샷을 생성할 수 있습니다.

MemoryDB 클러스터를 삭제할 때 최종 스냅샷을 만들려면 삭제 페이지에서 Yes 를 선택하고 5단계: 클러스터 삭제에서 스냅샷 이름을 지정합니다.

최종 스냅샷 생성(AWSCLI)

AWS CLI을(를) 사용하여 MemoryDB 클러스터를 삭제할 때 최종 스냅샷을 생성할 수 있습니다.

MemoryDB 클러스터를 삭제하는 경우

클러스터를 삭제할 때 최종 스냅샷을 생성하려면 다음 파라미터와 함께 `delete-cluster`AWS CLI작업을 사용합니다.

- `--cluster-name` - 삭제 중인 클러스터의 이름입니다.
- `--final-snapshot-name` - 최종 스냅샷의 이름입니다.

다음 코드는 클러스터 `myCluster`를 삭제할 때 최종 스냅샷 `bkup-20210515-final`을(를) 생성합니다.

Linux, macOS, Unix의 경우:

```
aws memorydb delete-cluster \
        --cluster-name myCluster \
        --final-snapshot-name bkup-20210515-final
```

Windows의 경우:

```
aws memorydb delete-cluster ^
        --cluster-name myCluster ^
        --final-snapshot-name bkup-20210515-final
```

자세한 내용은 AWS CLI 명령 참조의 delete-cluster를 참조하세요.

최종 스냅샷 생성(MemoryDB API)

MemoryDB API를 사용하여 MemoryDB 클러스터를 삭제할 때 최종 스냅샷을 생성할 수 있습니다.

MemoryDB 클러스터를 삭제하는 경우

최종 스냅샷을 만들려면 다음 파라미터와 함께 `DeleteCluster` MemoryDB API 작업을 사용하세요.

- `ClusterName` - 삭제 중인 클러스터의 이름입니다.
- `FinalSnapshotName` – 스냅샷의 이름입니다.

다음 MemoryDB API 작업에서는 클러스터 `myCluster`을(를) 삭제할 때 `bkup-20210515-final` 스냅샷을 생성합니다.

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DeleteCluster
    &ClusterName=myCluster
    &FinalSnapshotName=bkup-20210515-final
    &Version=2021-01-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210515T192317Z
    &X-Amz-Credential=<credential>
```

자세한 내용은 DeleteCluster 단원을 참조하세요.

스냅샷 설명

다음 절차는 스냅샷 목록을 표시하는 방법을 보여줍니다. 원한다면 특정 스냅샷의 세부 정보를 볼 수도 있습니다.

스냅샷 설명(콘솔)

를 사용하여 스냅샷을 표시하려면AWS Management 콘솔

1. 콘솔에 로그인합니다.
2. 왼쪽 탐색 창에서 Snapshots을 선택합니다.
3. 검색을 사용하여 수동, 자동 또는 모든 스냅샷을 필터링할 수 있습니다.
4. 특정 스냅샷의 세부 정보를 보려면 스냅샷 이름 왼쪽에 있는 라디오 버튼을 선택합니다. 작업을 선택한 다음 세부 정보 보기를 선택합니다.
5. 선택적으로 세부 정보 보기 페이지에서 복사, 복원 또는 삭제와 같은 추가 스냅샷 작업을 수행할 수 있습니다. 스냅샷에 태그를 추가할 수도 있습니다.

스냅샷 설명(AWSCLI)

스냅샷 목록을 표시하고 특정 스냅샷에 대한 세부 정보를 선택적으로 표시하려면 `describe-snapshots` CLI 작업을 사용하세요.

예시

다음 작업에서는 `--max-results` 파라미터를 사용하여 계정에 연결된 스냅샷을 20개까지 나열합니다. `--max-results` 파라미터를 생략하면 스냅샷이 50개까지 나열됩니다.

```
aws memorydb describe-snapshots --max-results 20
```

다음 작업에서는 `--cluster-name` 파라미터를 사용하여 클러스터 `my-cluster`에 연결된 스냅샷만 나열합니다.

```
aws memorydb describe-snapshots --cluster-name my-cluster
```

다음 작업에서는 `--snapshot-name` 파라미터를 사용하여 스냅샷 `my-snapshot`의 세부 정보를 표시합니다.

```
aws memorydb describe-snapshots --snapshot-name my-snapshot
```

자세한 내용은 [describe-snapshots](https://docs.aws.amazon.com/cli/latest/reference/memorydb/describe-snapshots.html)를 참조하세요.

스냅샷 설명 (MemoryDB API)

스냅샷 목록을 표시하려면 `DescribeSnapshots` 작업을 사용하세요.

예시

다음 작업에서는 `MaxResults` 파라미터를 사용하여 계정에 연결된 스냅샷을 20개까지 나열합니다. `MaxResults` 파라미터를 생략하면 스냅샷이 50개까지 나열됩니다.

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DescribeSnapshots
    &MaxResults=20
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

다음 작업에서는 `ClusterName` 파라미터를 사용하여 클러스터 `MyCluster`에 연결된 모든 스냅샷을 나열합니다.

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DescribeSnapshots
    &ClusterName=MyCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

다음 작업에서는 `SnapshotName` 파라미터를 사용하여 스냅샷 `MyBackup`의 세부 정보를 표시합니다.

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DescribeSnapshots
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &SnapshotName=MyBackup
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

자세한 내용은 DescribeSnapshots를 참조하세요.

스냅샷 복사

스냅샷을 자동으로 생성했건 수동으로 생성했건 스냅샷 복사본을 만들 수 있습니다. 스냅샷을 복사할 때 특별히 재정의하지 않는 한 원본과 동일한 KMS 암호화 키가 대상에 사용됩니다. 또한 스냅샷을 내보낼 수 있으므로 MemoryDB 외부에서 스냅샷에 액세스할 수 있습니다. 스냅샷을 내보내기 위한 지침은 [스냅샷 내보내기](#) 섹션을 참조하세요.

다음 절차에서는 스냅샷을 복사하는 방법을 보여줍니다.

스냅샷 복사(콘솔)

스냅샷을 복사하는 방법(console)

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/memorydb/ MemoryDB 콘솔을 엽니다.
2. 스냅샷 목록을 표시하려면 왼쪽 탐색 창에서 Snapshots을 선택합니다.
3. 스냅샷 목록에서 삭제할 스냅샷 왼쪽에 있는 버튼을 선택합니다.
4. 작업을 선택하고 복사를 선택합니다.
5. 스냅샷 복사 페이지에서 다음을 수행하세요.

    a. 새로운 스냅샷 이름 상자에 새로운 스냅샷 이름을 입력합니다.
    b. 선택 사항인 Target S3 Bucket 상자를 비워 둡니다. 이 필드는 스냅샷을 내보내기 위해서만 사용해야 하며 특수한 S3 권한을 필요로 합니다. 스냅샷 내보내기에 대한 정보는 [스냅샷 내보내기](#) 섹션을 참조하세요.
    c. 기본 AWS KMS암호화 키를 사용할지 아니면 사용자 지정 키를 사용할지 선택합니다. 자세한 내용은 [MemoryDB에서 전송 중 데이터 암호화(TLS)](#) 단원을 참조하십시오.
    d. 선택적으로 스냅샷 복사본에 태그를 추가할 수도 있습니다.
    e. 복사를 선택합니다.

스냅샷 복사(AWSCLI)

스냅샷을 복사하려면 `copy-snapshot` 작업을 사용합니다.

Parameters

- `--source-snapshot-name` – 복사할 스냅샷의 이름입니다.
- `--target-snapshot-name` - 스냅샷 복사본의 이름입니다.

- `--target-bucket`- 스냅샷 내보내기가 예약됩니다. 스냅샷 복사본을 만들 때 이 파라미터를 사용하지 마세요. 자세한 내용은 스냅샷 내보내기 단원을 참조하십시오.

다음 예제에서는 자동 스냅샷 복사본을 만듭니다.

Linux, macOS, Unix의 경우:

```
aws memorydb copy-snapshot \
    --source-snapshot-name automatic.my-primary-2021-03-27-03-15 \
    --target-snapshot-name my-snapshot-copy
```

Windows의 경우:

```
aws memorydb copy-snapshot ^
    --source-snapshot-name automatic.my-primary-2021-03-27-03-15 ^
    --target-snapshot-name my-snapshot-copy
```

자세한 내용은 copy-snapshot을 참조하세요.

스냅샷 복사(MemoryDB API)

스냅샷을 복사하려면 다음 파라미터와 함께 `copy-snapshot` 작업을 사용하세요.

Parameters

- `SourceSnapshotName` – 복사할 스냅샷의 이름입니다.
- `TargetSnapshotName` - 스냅샷 복사본의 이름입니다.
- `TargetBucket`- 스냅샷 내보내기가 예약됩니다. 스냅샷 복사본을 만들 때 이 파라미터를 사용하지 마세요. 자세한 내용은 스냅샷 내보내기 단원을 참조하십시오.

다음 예제에서는 자동 스냅샷 복사본을 만듭니다.

Example

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=CopySnapshot
    &SourceSnapshotName=automatic.my-primary-2021-03-27-03-15
    &TargetSnapshotName=my-snapshot-copy
    &SignatureVersion=4
```

```
    &SignatureMethod=HmacSHA256
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

자세한 내용은 CopySnapshot을 참조하세요.

스냅샷 내보내기

MemoryDB에서는 Amazon Simple Storage Service(Amazon S3) 버킷에 MemoryDB 스냅샷 내보내기가 지원되므로 MemoryDB 외부에서 스냅샷에 액세스할 수 있습니다. 내보낸 MemoryDB 스냅샷은 Valkey 및 오픈 소스 Redis OSS와 완벽하게 호환되며 적절한 버전 또는 도구를 사용하여 로드할 수 있습니다. MemoryDB 콘솔AWS CLI, 또는 MemoryDB API를 사용하여 스냅샷을 내보낼 수 있습니다.

다른 AWS리전에서 클러스터를 시작해야 하는 경우 스냅샷을 내보내는 것이 유용할 수 있습니다. 데이터를 한 AWS리전으로 내보내고 .rdb 파일을 새 AWS리전으로 복사한 다음 새 클러스터가 사용을 통해 채워질 때까지 기다리는 대신 해당 .rdb 파일을 사용하여 새 클러스터를 시드할 수 있습니다. 새 클러스터 시드에 대한 자세한 내용은 외부에서 생성된 스냅샷으로 새 클러스터 시드 섹션을 참조하세요. 오프라인 처리를 위해 .rdb 파일을 사용하기 위해 클러스터의 데이터를 내보낼 수도 있습니다.

> **Important**
>
> - MemoryDB 스냅샷과 이를 복사하려는 Amazon S3 버킷은 동일한 AWS리전에 있어야 합니다.
>
>   Amazon S3 버킷으로 복사된 스냅샷이 암호화되긴 하지만 스냅샷을 저장하려는 Amazon S3 버킷에 대한 액세스 권한을 다른 사람에게 부여하지 않는 것이 좋습니다.
> - 데이터 계층화를 사용하는 클러스터는 Amazon S3로 스냅샷 내보내기가 지원되지 않습니다. 자세한 내용은 데이터 계층화 단원을 참조하십시오.

스냅샷을 Amazon S3 버킷으로 내보내려면 먼저 스냅샷과 동일한 AWS리전에 Amazon S3 버킷이 있어야 합니다. 버킷에 MemoryDB 액세스 권한을 부여합니다. 처음 두 단계에서는 이 작업을 수행할 방법을 보여줍니다.

> **Warning**
>
> 다음 시나리오에서는 사용자가 원하지 않을 수도 있는 방법으로 데이터를 노출합니다.
>
> - 스냅샷을 내보낸 Amazon S3 버킷에 대한 액세스 권한이 다른 사람에게 있는 경우.
>
>   스냅샷에 대한 액세스를 제어하려면 데이터에 대한 액세스를 허용할 사람에게만 Amazon S3 버킷에 대한 액세스를 허용합니다. Amazon S3 버킷에 대한 액세스 관리에 대한 자세한 내용은 Amazon S3 개발자 안내서의 액세스 관리를 참조하세요.
> - 다른 사람이 CopySnapshot API 작업을 사용할 권한을 가진 경우

CopySnapshot API 작업을 사용할 권한을 가진 사용자나 그룹이 자체 Amazon S3 버킷을 생성하고 스냅샷을 이 버킷에 복사할 수 있습니다. 스냅샷에 대한 액세스를 제어하려면 AWS Identity and Access Management(IAM) 정책을 사용하여 CopySnapshot API를 사용할 수 있는 사용자를 제어합니다. IAM을 사용하여 MemoryDB API 작업 사용을 제어하는 방법에 대한 자세한 내용은 MemoryDB 사용 설명서의 MemoryDB의 ID 및 액세스 관리 섹션을 참조하세요.

주제

1단계: Amazon S3 버킷 생성

다음 절차에서는 Amazon S3 콘솔을 사용하여 MemoryDB 스냅샷을 내보내고 저장할 Amazon S3 버킷을 생성합니다.

Amazon S3 버킷을 생성하려면

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/s3/ Amazon S3 콘솔을 엽니다.
2. 버킷 생성을 선택합니다.
3. [Create a Bucket - Select a Bucket Name and Region]에서 다음을 수행합니다.

    a. 버킷 이름에서 Amazon S3 버킷의 이름을 입력합니다.

    b. 리전 목록에서 Amazon S3 버킷의 AWS리전을 선택합니다. 이 AWS리전은 내보내려는 MemoryDB 스냅샷과 동일한 AWS리전이어야 합니다.

    c. 생성(Create)을 선택합니다.

Amazon S3 버킷 생성에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 버킷 생성을 참조하세요.

2단계: Amazon S3 버킷에 대한 MemoryDB 액세스 권한 부여

AWS2019년 3월 20일 이전에 도입된 리전은 기본적으로 활성화됩니다. 이러한 AWS리전에서 즉시 작업을 시작할 수 있습니다. 2019년 3월 20일 이후에 도입된 리전은 기본적으로 비활성 상태입니다. 사용하려면 먼저 AWS 리전 관리에 설명된 대로 리전을 활성화하거나 옵트인해야 합니다.

AWS리전의 S3 버킷에 대한 MemoryDB 액세스 권한 부여

AWS리전의 Amazon S3 버킷에 대한 적절한 권한을 생성하려면 다음 단계를 수행합니다.

S3 bucket 버킷에 MemoryDB 액세스 권한을 부여하려면

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/s3/ Amazon S3 콘솔을 엽니다.
2. 스냅샷을 복사할 Amazon S3 버킷 이름을 선택합니다. 1단계: Amazon S3 버킷 생성에서 생성한 S3 버킷이어야 합니다.
3. 권한 탭을 선택한 후 권한에서 버킷 정책을 선택합니다.
4. 정책을 업데이트하여 MemoryDB에 작업을 수행하는 데 필요한 권한을 부여합니다.
   - `[ "Service" : "region-full-name.memorydb-snapshot.amazonaws.com" ]`을 `Principal`에 추가합니다.
   - Amazon S3 버킷으로 스냅샷을 내보내는 데 필요한 다음 권한을 추가합니다.
     - `"s3:PutObject"`
     - `"s3:GetObject"`
     - `"s3:ListBucket"`
     - `"s3:GetBucketAcl"`
     - `"s3:ListMultipartUploadParts"`
     - `"s3:ListBucketMultipartUploads"`

   다음은 업데이트된 정책의 예입니다.

   JSON

```
{
    "Version":"2012-10-17",
    "Id": "Policy15397346",
    "Statement": [
```

```
        {
            "Sid": "Stmt15399483",
            "Effect": "Allow",
            "Principal": {
                "Service": "aws-region.memorydb-snapshot.amazonaws.com"
            },
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketAcl",
                "s3:ListMultipartUploadParts",
                "s3:ListBucketMultipartUploads"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

3단계: MemoryDB 스냅샷 내보내기

이제 S3 버킷을 생성하고 MemoryDB에 액세스할 수 있는 권한을 부여했습니다. S3 객체 소유권을 ACL이 활성화 - 버킷 소유자 우대로 변경합니다. 다음으로 MemoryDB 콘솔, AWSCLI 또는 MemoryDB API를 사용하여 스냅샷을 내보낼 수 있습니다. 다음 예제에서는 아래의 추가 S3 관련 IAM 권한이 있다고 가정합니다.

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [{
  "Effect": "Allow",
  "Action": [
   "s3:GetBucketLocation",
   "s3:ListAllMyBuckets",
   "s3:PutObject",
   "s3:GetObject",
   "s3:DeleteObject",
```

```
    "s3:ListBucket"
  ],
  "Resource": "arn:aws:s3:::*"
 }]
}
```

MemoryDB 스냅샷 내보내기(콘솔)

다음 프로세스에서는 MemoryDB 외부에서 액세스할 수 있도록 MemoryDB 콘솔을 사용하여 Amazon S3 버킷에 스냅샷을 내보냅니다. Amazon S3 버킷은 MemoryDB 스냅샷과 동일한 AWS리전에 있어야 합니다.

Amazon S3 버킷으로 MemoryDB 스냅샷 내보내기

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/memorydb/ MemoryDB 콘솔을 엽니다.
2. 스냅샷 목록을 표시하려면 왼쪽 탐색 창에서 Snapshots을 선택합니다.
3. 스냅샷 목록에서 내보내려는 스냅샷 이름의 왼쪽에 있는 라디오 버튼을 선택합니다.
4. 복사를 선택합니다.
5. 백업 복사본을 생성에서 다음 작업을 수행하세요.

   a. 새로운 스냅샷 이름 상자에 새로운 스냅샷 이름을 입력합니다.

      이름은 1~1,000자여야 하며 UTF-8 인코딩일 수 있습니다.

      MemoryDB에서는 여기에 입력하는 값에 샤드 식별자 및 `.rdb`를 추가합니다. 예를 들어 `my-exported-snapshot`을 입력하면 MemoryDB에서 `my-exported-snapshot-0001.rdb`를 생성합니다.

   b. 대상 S3 위치 목록에서 스냅샷을 복사할 Amazon S3 버킷(1단계: Amazon S3 버킷 생성에서 생성한 버킷)을 선택합니다.

      대상 S3 위치는 내보내기 프로세스가 성공하려면 다음 권한이 있는 스냅샷 AWS리전의 Amazon S3 버킷이어야 합니다.

      • 객체 액세스 - 읽기 및 쓰기.
      • 권한 액세스 - 읽기.

자세한 내용은 2단계: Amazon S3 버킷에 대한 MemoryDB 액세스 권한 부여 단원을 참조하십시오.

c. 복사를 선택합니다.

> **Note**
>
> MemoryDB에서 스냅샷을 내보내는 데 필요한 권한이 S3 버킷에 없으면 다음 오류 메시지 중 하나를 받습니다. 2단계: Amazon S3 버킷에 대한 MemoryDB 액세스 권한 부여로 돌아가 지정한 권한을 추가하고 스냅샷 내보내기를 다시 시도하세요.
>
> - S3 버킷에 대한 READ 권한이 MemoryDB에 부여되지 않았습니다.
>
>   해결 방법: 버킷에 대한 읽기 권한을 추가합니다.
> - S3 버킷에 대한 WRITE 권한이 MemoryDB에 부여되지 않았습니다.
>
>   해결 방법: 버킷에 대한 쓰기 권한을 추가합니다.
> - S3 버킷에 대한 READ_ACP 권한이 MemoryDB에 부여되지 않았습니다.
>
>   해결 방법: 버킷에 대한 권한 액세스로 [Read]를 추가합니다.

스냅샷을 다른 AWS리전에 복사하려면 Amazon S3를 사용하여 복사합니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 객체 복사를 참조하세요.

MemoryDB 스냅샷 내보내기(AWSCLI)

다음 파라미터와 함께 `copy-snapshot` CLI 작업을 사용하여 Amazon S3 버킷에 스냅샷을 내보냅니다.

Parameters

- `--source-snapshot-name` – 복사할 스냅샷의 이름입니다.
- `--target-snapshot-name` - 스냅샷 복사본의 이름입니다.

  이름은 1~1,000자여야 하며 UTF-8 인코딩일 수 있습니다.

MemoryDB에서는 여기에 입력하는 값에 샤드 식별자 및 `.rdb`를 추가합니다. 예를 들어 `my-exported-snapshot`을 입력하면 MemoryDB에서 `my-exported-snapshot-0001.rdb`를 생성합니다.

- `--target-bucket` - 스냅샷을 내보낼 Amazon S3 버킷의 이름입니다. 지정한 버킷에 스냅샷 복사본이 생성됩니다.

  내보내기 프로세스가 성공하려면가 다음 권한이 AWS있는 스냅샷 리전의 Amazon S3 버킷이어야 `--target-bucket` 합니다.
  - 객체 액세스 - 읽기 및 쓰기.
  - 권한 액세스 - 읽기.

  자세한 내용은 2단계: Amazon S3 버킷에 대한 MemoryDB 액세스 권한 부여 단원을 참조하십시오.

다음 작업은 amzn-s3-demo-bucket에 스냅샷을 복사합니다.

Linux, macOS, Unix의 경우:

```
aws memorydb copy-snapshot \
    --source-snapshot-name automatic.my-primary-2021-06-27-03-15 \
    --target-snapshot-name my-exported-snapshot \
    --target-bucket amzn-s3-demo-bucket
```

Windows의 경우:

```
aws memorydb copy-snapshot ^
    --source-snapshot-name automatic.my-primary-2021-06-27-03-15 ^
    --target-snapshot-name my-exported-snapshot ^
    --target-bucket amzn-s3-demo-bucket
```

> **Note**
>
> MemoryDB에서 스냅샷을 내보내는 데 필요한 권한이 S3 버킷에 없으면 다음 오류 메시지 중 하나를 받습니다. 2단계: Amazon S3 버킷에 대한 MemoryDB 액세스 권한 부여로 돌아가 지정한 권한을 추가하고 스냅샷 내보내기를 다시 시도하세요.
>
> - S3 버킷에 대한 READ 권한이 MemoryDB에 부여되지 않았습니다.
>
>   해결 방법: 버킷에 대한 읽기 권한을 추가합니다.

- S3 버킷에 대한 WRITE 권한이 MemoryDB에 부여되지 않았습니다.

  해결 방법: 버킷에 대한 쓰기 권한을 추가합니다.
- S3 버킷에 대한 READ_ACP 권한이 MemoryDB에 부여되지 않았습니다.

  해결 방법: 버킷에 대한 권한 액세스로 [Read]를 추가합니다.

자세한 내용은 AWS CLI 명령 참조의 `copy-snapshot` 섹션을 참조하십시오.

스냅샷을 다른 AWS리전에 복사하려면 Amazon S3 복사를 사용합니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 객체 복사를 참조하세요.

MemoryDB 스냅샷 내보내기(MemoryDB API)

이러한 파라미터와 함께 `CopySnapshot` API 작업을 사용하여 Amazon S3 버킷에 스냅샷을 내보냅니다.

Parameters

- `SourceSnapshotName` – 복사할 스냅샷의 이름입니다.
- `TargetSnapshotName` - 스냅샷 복사본의 이름입니다.

  이름은 1~1,000자여야 하며 UTF-8 인코딩일 수 있습니다.

  MemoryDB에서는 여기에 입력하는 값에 샤드 식별자 및 `.rdb`을(를) 추가합니다. 예를 들어 `my-exported-snapshot`을 입력하면 `my-exported-snapshot-0001.rdb`가 생깁니다.
- `TargetBucket` - 스냅샷을 내보낼 Amazon S3 버킷의 이름입니다. 지정한 버킷에 스냅샷 복사본이 생성됩니다.

  내보내기 프로세스가 성공하려면가 다음 권한이 AWS있는 스냅샷 리전의 Amazon S3 버킷이어야 `TargetBucket` 합니다.

  - 객체 액세스 - 읽기 및 쓰기.
  - 권한 액세스 - 읽기.

  자세한 내용은 2단계: Amazon S3 버킷에 대한 MemoryDB 액세스 권한 부여 단원을 참조하십시오.

다음 예제에서는 Amazon S3 버킷 `amzn-s3-demo-bucket`에 자동 스냅샷 복사본을 만듭니다.

Example

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=CopySnapshot
    &SourceSnapshotName=automatic.my-primary-2021-06-27-03-15
    &TargetBucket=&example-s3-bucket;
    &TargetSnapshotName=my-snapshot-copy
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

> **Note**
>
> MemoryDB에서 스냅샷을 내보내는 데 필요한 권한이 S3 버킷에 없으면 다음 오류 메시지 중 하나를 받습니다. 2단계: Amazon S3 버킷에 대한 MemoryDB 액세스 권한 부여로 돌아가 지정한 권한을 추가하고 스냅샷 내보내기를 다시 시도하세요.
>
> - S3 버킷에 대한 READ 권한이 MemoryDB에 부여되지 않았습니다.
>
>   해결 방법: 버킷에 대한 읽기 권한을 추가합니다.
> - S3 버킷에 대한 WRITE 권한이 MemoryDB에 부여되지 않았습니다.
>
>   해결 방법: 버킷에 대한 쓰기 권한을 추가합니다.
> - S3 버킷에 대한 READ_ACP 권한이 MemoryDB에 부여되지 않았습니다.
>
>   해결 방법: 버킷에 대한 권한 액세스로 [Read]를 추가합니다.

자세한 내용은 CopySnapshot을 참조하세요.

스냅샷을 다른 AWS리전에 복사하려면 Amazon S3 복사를 사용하여 내보낸 스냅샷을 다른 AWS리전의 Amazon S3 버킷에 복사합니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 객체 복사를 참조하세요.

스냅샷에서 복원

언제든지 MemoryDB 또는 ElastiCache(Redis OSS) .rdb 스냅샷 파일에서 새 클러스터로 데이터를 복원할 수 있습니다.

MemoryDB 복원 프로세스는 다음을 지원합니다.

- ElastiCache(Redis OSS)에서 생성한 하나 이상의 .rdb 스냅샷 파일에서 MemoryDB 클러스터로의 마이그레이션입니다.

  .rdb 파일을 S3에 배치해야 복원할 수 있습니다.
- 스냅샷 파일 생성에 사용한 클러스터의 샤드 수와 다른 새 클러스터의 샤드 수 지정
- 더 크거나 작은 새 클러스터의 다른 노드 유형 지정. 더 작은 노드 유형으로 규모를 조정하는 경우 새로운 노드 유형에 충분한 메모리가 있어 데이터와 엔진 오버헤드를 수용할 수 있어야 합니다.
- 스냅샷 파일 생성에 사용한 클러스터에서와 다른 새 MemoryDB 클러스터의 슬롯 구성입니다.

> **Important**
>
> - MemoryDB 클러스터는 여러 데이터베이스를 지원하지 않습니다. 따라서 MemoryDB로 복원하는 경우, .rdb 파일이 데이터베이스를 두 개 이상 참조하면 복원에 실패합니다.
> - 데이터 계층화(예: r6gd 노드 유형)를 사용하는 클러스터에서 데이터 계층화를 사용하지 않는 클러스터(예: r6g 노드 유형)로 스냅샷을 복원할 수 없습니다.

스냅샷에서 클러스터를 복원할 때 변경 여부를 선택할 수 있습니다. 이러한 선택은 MemoryDB 콘솔을 사용하여 복원할 때 클러스터 복원(Restore Cluster) 대화 상자에서 할 수 있습니다. AWS CLI또는 MemoryDB API를 사용하여 복원할 때 파라미터 값을 설정하여 이러한 선택을 합니다.

복원 작업 중에 MemoryDB가 새 클러스터를 만든 후 스냅샷 파일의 데이터로 클러스터를 채웁니다. 이 프로세스가 완료되면 클러스터가 워밍업되어 요청을 수락할 준비가 됩니다.

> **Important**
>
> 계속하기 전에 복원할 클러스터의 스냅샷을 생성해야 합니다. 자세한 내용은 수동 스냅샷 생성 단원을 참조하십시오.
> 외부에서 생성된 스냅샷에서 복원하려면 외부에서 생성된 스냅샷으로 새 클러스터 시드 섹션을 참조하세요.

다음 절차에서는 MemoryDB 콘솔AWS CLI, 또는 MemoryDB API를 사용하여 스냅샷을 새 클러스터로 복원하는 방법을 보여줍니다.

스냅샷에서 복원(콘솔)

새 클러스터로 스냅샷을 복원하려면(콘솔)

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/memorydb/ MemoryDB 콘솔을 엽니다.
2. 탐색 창에서 스냅샷(Snapshots)를 선택합니다.
3. 스냅샷 목록에서 복원할 스냅샷 이름 옆의 버튼을 선택합니다.
4. 작업을 선택한 후 복원을 선택합니다.
5. 클러스터 구성에서 다음을 입력합니다.

   a. 클러스터 ID - 필수입니다. 새 클러스터의 이름입니다.
   b. 설명 - 선택 사항 새 클러스터의 설명입니다.
6. 서브넷 그룹 섹션을 완료하세요.

   • 서브넷 그룹에서 새 서브넷 그룹을 만들거나 사용 가능한 목록에서 이 클러스터에 적용할 기존 서브넷 그룹을 선택합니다. 새로 만드는 경우:

     • 이름을 입력합니다.
     • 설명을 입력합니다.
     • 다중 AZ를 활성화한 경우, 서브넷 그룹에는 서로 다른 가용 영역에 상주하는 서브넷이 두 개 이상 있어야 합니다. 자세한 내용은 서브넷 및 서브넷 그룹 단원을 참조하십시오.
     • 새 서브넷 그룹을 만들고 기존 VPC가 없는 경우, VPC를 생성하라는 메시지가 표시됩니다. 자세한 내용은 Amazon VPC 사용 설명서의 Amazon VPC란 무엇인가요?를 참조하세요.
7. 클러스터 설정 단원을 완료합니다.

   a. Valkey 버전 호환성 또는 Redis OSS 버전 호환성을 위해 기본값인 `6.0`을 그대로 사용합니다.
   b. 포트의 경우, 기본 포트인 6379를 그대로 사용하거나, 다른 포트를 사용해야 하는 이유가 있는 경우 포트 번호를 입력합니다.
   c. 파라미터 그룹의 경우, `default.memorydb-redis6` 파라미터 그룹을 수락합니다.

      파라미터 그룹은 클러스터의 런타임 파라미터를 제어합니다. 파라미터 그룹에 대한 자세한 정보는 엔진별 파라미터 단원을 참조하세요.

d. 노드 유형에서 원하는 노드 유형 값(관련 메모리 크기 포함)을 선택합니다.

   r6gd 패밀리의 노드 유형을 선택하는 경우, 클러스터에 데이터 계층화가 자동으로 활성화됩니다. 자세한 내용은 데이터 계층화 단원을 참조하십시오.

e. 샤드 수에서 이 클러스터에 사용할 샤드 수를 선택합니다.

   클러스터의 샤드 수를 동적으로 변경할 수 있습니다. 자세한 내용은 MemoryDB 클러스터 크기 조정 단원을 참조하십시오.

f. 샤드당 복제본에서 각 샤드에 포함할 읽기 전용 복제본 노드 수를 선택합니다.

   다음과 같은 제한 사항이 있습니다.

   - 다중 AZ를 활성화한 경우 샤드당 복제본이 하나 이상 있어야 합니다.
   - 콘솔을 사용하여 클러스터를 생성할 때 샤드마다 복제본 수가 동일합니다.

g. 다음을 선택합니다.

h. 고급 설정 섹션을 완료하세요.

   i. 보안 그룹에서 이 클러스터에 사용할 보안 그룹을 선택합니다. 보안 그룹은 클러스터에 대한 네트워크 액세스를 제어하는 방화벽 역할을 합니다. VPC의 기본 보안 그룹을 사용하거나 새 보안 그룹을 만들 수 있습니다.

      보안 그룹에 대한 자세한 정보는 Amazon VPC 사용 설명서의 VPC의 보안 그룹을 참조하세요.

   ii. 데이터는 다음과 같은 방식으로 암호화됩니다.

      - 저장된 데이터 암호화 - 디스크에 저장된 데이터 암호화를 활성화합니다. 자세한 정보는 저장된 데이터 암호화를 참조하세요.

        > **Note**
        >
        > 고객 관리형 AWSKMS 키를 선택하고 키를 선택하여 다른 암호화 키를 제공할 수 있습니다.

      - 전송 중 데이터 암호화 – 전송 데이터 암호화를 활성화합니다. 이는 기본값으로 사용 설정되어 있습니다. 자세한 정보는 전송 중 데이터 암호화를 참조하세요.

암호화를 선택하지 않으면 "오픈 액세스"라는 개방형 액세스 제어 목록이 기본 사용자와 함께 생성됩니다. 자세한 내용은 액세스 제어 목록(ACL)을 사용하여 사용자 인증 단원을 참조하십시오.

iii. 스냅샷의 경우, 스냅샷 보존 기간과 스냅샷 기간을 선택적으로 지정할 수 있습니다. 기본적으로 자동 스냅샷 활성화가 선택됩니다.

iv. 유지 관리 창의 경우, 선택적으로 유지 관리 기간을 지정할 수 있습니다. 유지 관리 기간은 MemoryDB가 클러스터의 시스템 유지 관리를 예약하는 시간이며 일반적으로 매주 한 시간입니다. MemoryDB에서 유지 관리 기간의 요일과 시간을 선택하도록 허용하거나(기본 설정 없음) 요일 시간 및 기간을 직접 선택할 수 있습니다(유지 관리 기간 지정). [Specify maintenance window]를 선택할 경우 목록에서 유지 관리 기간의 [Start day], [Start time] 및 [Duration](시간)을 선택합니다. 모든 시간은 UCT 시간입니다.

   자세한 내용은 유지 관리 관리 중 단원을 참조하십시오.

v. 알림에 대해 기존의 Amazon Simple Notification Service(Amazon SNS) 항목을 선택하거나 수동 ARN 입력을 선택하고 Amazon 리소스 이름(ARN) 항목을 입력합니다. Amazon SNS를 통해 인터넷에 연결된 스마트 디바이스에 알림을 푸시할 수 있습니다. 기본적으로 알림이 비활성화됩니다. 자세한 내용은 https://aws.amazon.com/sns/를 참조하세요.

i. 태그의 경우 선택적으로 태그를 적용하여 클러스터를 검색 및 필터링하거나 AWS비용을 추적할 수 있습니다.

j. 입력 및 선택한 내용을 모두 검토한 다음 필요한 내용을 수정합니다. 준비가 되면 클러스터 생성을 선택하여 클러스터를 시작하거나 취소를 선택해 작업을 취소합니다.

클러스터 상태가 사용 가능이 되면 클러스터에 EC2 액세스 권한을 부여하고 클러스터에 연결하며 사용할 수 있습니다. 자세한 내용은 3단계: 클러스터에 대한 액세스 허가 및 4단계: 클러스터에 연결 섹션을 참조하세요.

> **Important**
>
> 클러스터를 사용할 수 있게 되면 클러스터를 적극 사용하지 않더라도 클러스터가 활성화되어 있는 매 시간 또는 60분 미만 단위로 비용이 청구됩니다. 이 클러스터의 요금 발생을 중지하려면 클러스터를 삭제해야 합니다. 5단계: 클러스터 삭제을(를) 참조하세요.

스냅샷에서 복원(AWSCLI)

`create-cluster` 작업을 사용할 때는 파라미터 `--snapshot-name` 또는 `--snapshot-arns`을 포함하여 스냅샷의 데이터로 새로운 클러스터를 시드해야 합니다.

자세한 내용은 다음을 참조하세요.

- 클러스터 생성(AWSCLI)MemoryDB 사용 설명서에서
- AWS CLI명령 참조의 create-cluster

스냅샷에서 복원(MemoryDB API)

MemoryDB API 작업 `CreateCluster`을 사용하여 MemoryDB 스냅샷을 복원할 수 있습니다.

`CreateCluster` 작업을 사용할 때는 파라미터 `SnapshotName` 또는 `SnapshotArns`을 포함하여 스냅샷의 데이터로 새로운 클러스터를 시드해야 합니다.

자세한 내용은 다음을 참조하세요.

- 클러스터 생성(MemoryDB API)MemoryDB 사용 설명서에서
- MemoryDB API 참조의 CreateCluster

외부에서 생성된 스냅샷으로 새 클러스터 시드

새 MemoryDB 클러스터를 만들 때 Valkey 또는 Redis OSS .rdb 스냅샷 파일의 데이터로 이 클러스터를 시드할 수 있습니다.

MemoryDB 스냅샷 또는 ElastiCache(Redis OSS) 스냅샷에서 새 MemoryDB 클러스터를 시드하려면 스냅샷에서 복원 섹션을 참조하세요.

.rdb 파일을 사용하여 새 MemoryDB 클러스터를 시드할 때 다음을 수행할 수 있습니다.

- 새로운 클러스터의 샤드 수를 지정합니다. 이 숫자는 스냅샷 파일을 생성하는 데 사용된 클러스터의 샤드 수와 다를 수 있습니다.
- 스냅샷을 만든 클러스터에 사용된 것보다 크거나 작은 새 클러스터의 다른 노드 유형을 지정합니다. 더 작은 노드 유형으로 규모를 조정하는 경우 새로운 노드 유형에 충분한 메모리가 있어 데이터와 엔진 오버헤드를 수용할 수 있어야 합니다.

> **Important**
>
> - Redis 스냅샷 데이터가 노드의 리소스를 초과하지 않아야 합니다.
>
>   스냅샷이 너무 크면 결과 클러스터의 상태는 `restore-failed`이(가) 됩니다. 이 경우 클러스터를 삭제하고 다시 시작해야 합니다.
>
>   노드 유형 및 사양의 전체 목록은 MemoryDB 노드 유형별 파라미터을(를) 참조하세요.
> - Amazon S3 서버 측 암호화(SSE-S3)만으로 .rdb 파일을 암호화할 수 있습니다. 자세한 내용은 서버 측 암호화를 사용하여 데이터 보호를 참조하세요.

1단계: 외부 클러스터에 스냅샷 생성

MemoryDB 클러스터를 시드할 스냅샷을 만드는 방법

1. 기존의 Valkey 또는 Redis OSS 인스턴스에 연결합니다.
2. `BGSAVE` 또는 `SAVE` 작업을 실행하여 스냅샷을 생성합니다. .rdb 파일의 위치를 메모합니다.

   `BGSAVE`는 비동기식이며 처리하는 동안 다른 클라이언트를 차단하지 않습니다. 자세한 내용은 BGSAVE를 참조하세요.

SAVE는 동기식이며 마칠 때까지 다른 프로세스를 차단합니다. 자세한 내용은 SAVE를 참조하세요.

스냅샷 생성에 대한 추가 정보는 지속성을 참조하세요.

2단계: Amazon S3 버킷 및 폴더 생성

스냅샷 파일을 만들었으면 Amazon S3 버킷에 있는 폴더에 업로드해야 합니다. 그러려면 먼저 Amazon S3 버킷과 버킷 내의 폴더가 있어야 합니다. 적절한 권한을 가진 Amazon S3 버킷과 폴더가 이미 있으면 3단계: Amazon S3에 스냅샷 업로드 섹션으로 건너뛸 수 있습니다.

Amazon S3 버킷을 생성하려면

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/s3/ Amazon S3 콘솔을 엽니다.
2. Amazon Simple Storage Service 사용 설명서에서 버킷 생성의 Amazon S3 버킷 생성 지침을 따릅니다.

   Amazon S3 버킷의 이름은 DNS를 준수해야 합니다. 그렇지 않으면 MemoryDB가 스냅샷 파일에 액세스할 수 없습니다. DNS 준수 규칙은 다음과 같습니다.

   - 이름은 3자 이상, 63자 이하여야 합니다.
   - 이름은 마침표(.)로 구분된 일련의 레이블(1개 이상)이어야 합니다. 여기서 각 레이블은 다음과 같아야 합니다.
     - 소문자 또는 숫자로 시작합니다.
     - 소문자 또는 숫자로 끝납니다.
     - 소문자, 숫자 및 대시만 포함합니다.
   - 이름에는 IP 주소 형식(예: 192.0.2.0)을 사용할 수 없습니다.

   새 MemoryDB 클러스터와 동일한 AWS리전에 Amazon S3 버킷을 생성하는 것이 좋습니다. 이 방법은 MemoryDB가 Amazon S3에서 .rdb 파일을 읽을 때 최고의 데이터 전송 속도를 경험할 수 있도록 합니다.

> **Note**
>
> 데이터를 최대한 안전하게 유지하려면 Amazon S3 버킷에 대한 권한을 최대한 제한적으로 설정합니다. 또한 새 MemoryDB 클러스터를 시드하는 데 버킷과 버킷의 콘텐츠를 사용하기 위해 권한이 계속 필요합니다.

Amazon S3 버킷에 폴더를 추가하려면

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/s3/ Amazon S3 콘솔을 엽니다.
2. .rdb 파일을 업로드할 버킷 이름을 선택합니다.
3. 폴더 생성을 선택합니다.
4. 새 폴더의 이름을 입력합니다.
5. 저장을 선택합니다.

   버킷 이름과 폴더 이름을 모두 메모합니다.

3단계: Amazon S3에 스냅샷 업로드

이제 1단계: 외부 클러스터에 스냅샷 생성에서 생성한 .rdb 파일을 업로드합니다. 2단계: Amazon S3 버킷 및 폴더 생성에서 생성한 Amazon S3 버킷과 폴더로 업로드합니다. 객체 업로드에 대한 자세한 내용은 Uploading objects 단원을 참조하세요. 2단계와 3단계 사이에 생성된 폴더 이름을 선택합니다.

.rdb 파일을 Amazon S3 폴더에 업로드하려면

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/s3/ Amazon S3 콘솔을 엽니다.
2. 2단계에서 만든 Amazon S3 버킷 이름을 선택합니다.
3. 2단계에서 만든 폴더 이름을 선택합니다.
4. 업로드를 선택합니다.
5. Add Files를 선택합니다.
6. 업로드할 파일을 찾아 선택합니다. 파일을 여러 개 선택하려면 Ctrl 키를 누른 상태로 각 파일 이름을 선택합니다.
7. Open을 선택합니다.

8. Upload 페이지에서 정확한 파일 이름이 표시되는지 확인하고 Upload를 선택합니다.

.rdb 파일에 대한 경로를 기록합니다. 예를 들어 버킷 이름이 `amzn-s3-demo-bucket`이고 경로가 `myFolder/redis.rdb`이면 `amzn-s3-demo-bucket/myFolder/redis.rdb`를 입력합니다. 이 스냅샷의 데이터로 새 클러스터를 시드하려면 이 경로가 필요합니다.

자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [버킷 이름 지정 규칙]을 참조하세요.

4단계: MemoryDB에 .rdb 파일에 대한 읽기 액세스 부여

AWS2019년 3월 20일 이전에 도입된 리전은 기본적으로 활성화됩니다. 이러한 AWS리전에서 즉시 작업을 시작할 수 있습니다. 2019년 3월 20일 이후에 도입된 리전은 기본적으로 비활성 상태입니다. 사용하려면 먼저 [AWS 리전 관리]에 설명된 대로 리전을 활성화하거나 옵트인해야 합니다.

MemoryDB에 .rdb 파일에 대한 읽기 액세스 부여

MemoryDB에 스냅샷 파일에 대한 읽기 액세스 권한 부여

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/s3/ Amazon S3 콘솔을 엽니다.
2. .rdb 파일이 포함된 S3 버킷 이름을 선택합니다.
3. .rdb 파일이 포함된 폴더 이름을 선택합니다.
4. .rdb 스냅샷 파일 이름을 선택합니다. 선택한 파일 이름은 페이지 맨 위의 탭 위에 나타납니다.
5. 권한 탭을 선택합니다.
6. 권한(Permissions)에서 버킷 정책(Bucket policy)을 선택한 다음 편집(Edit)을 선택합니다.
7. 정책을 업데이트하여 MemoryDB에 작업을 수행하는 데 필요한 권한을 부여합니다.
   - `[ "Service" : "`*`region-full-name`*`.memorydb-snapshot.amazonaws.com" ]`을 `Principal`에 추가합니다.
   - Amazon S3 버킷으로 스냅샷을 내보내는 데 필요한 다음 권한을 추가합니다.
     - `"s3:GetObject"`
     - `"s3:ListBucket"`
     - `"s3:GetBucketAcl"`

   다음은 업데이트된 정책의 예입니다.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "Policy15397346",
    "Statement": [
        {
            "Sid": "Stmt15399483",
            "Effect": "Allow",
            "Principal": {
                "Service": "us-east-1.memorydb-snapshot.amazonaws.com"
            },
            "Action": [
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketAcl"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/snapshot1.rdb",
                "arn:aws:s3:::amzn-s3-demo-bucket/snapshot2.rdb"
            ]
        }
    ]
}
```

8. 저장을 선택합니다.

5단계: .rdb 파일 데이터로 MemoryDB 클러스터 시드

이제 MemoryDB 클러스터를 생성하고 .rdb 파일의 데이터로 클러스터를 시드할 수 있습니다. 클러스터를 생성하려면 MemoryDB 클러스터 생성의 지시를 따릅니다.

Amazon S3에 업로드한 스냅샷의 위치를 MemoryDB에 알리는 데 사용하는 방법은 클러스터 생성에 사용한 방법에 따라 결정됩니다.

.rdb 파일 데이터로 MemoryDB 클러스터 시드

- MemoryDB 콘솔 사용

엔진을 선택한 후 고급 설정 섹션을 확장하고 클러스터로 데이터 가져오기를 찾습니다. RDB 파일 시드 S3 위치 상자에 파일의 Amazon S3 경로를 입력합니다. .rdb 파일이 여러 개 있으면 쉼표로 구분된 목록에 각 파일의 경로를 입력합니다. Amazon S3 경로는 `*amzn-s3-demo-bucket*/*myFolder*/*myBackupFilename*.rdb`처럼 표시될 수 있습니다.

- 사용AWS CLI

  `create-cluster` 또는 `create-cluster` 작업을 사용하는 경우 `--snapshot-arns` 파라미터를 사용하여 각 .rdb 파일의 정규화된 ARN을 지정합니다. 예를 들어 `arn:aws:s3:::*amzn-s3-demo-bucket*/*myFolder*/*myBackupFilename*.rdb`입니다. ARN은 Amazon S3에 저장한 스냅샷 파일로 확인되어야 합니다.
- MemoryDB API 사용

  `CreateCluster` 또는 `CreateCluster` MemoryDB API 작업을 사용하는 경우, 파라미터 를 사용하여 각 .rdb 파일의 정규화된 ARN을 지정합니다. 예를 들어 `arn:aws:s3:::*amzn-s3-demo-bucket*/*myFolder*/*myBackupFilename*.rdb`입니다. ARN은 Amazon S3에 저장한 스냅샷 파일로 확인되어야 합니다.

클러스터 생성 프로세스 중에 스냅샷의 데이터가 클러스터에 쓰여집니다. MemoryDB 이벤트 메시지를 보면서 프로세스를 모니터링할 수 있습니다. 그러려면 MemoryDB 콘솔 화면에서 Events(이벤트)를 선택합니다. 또한AWSMemoryDB 명령줄 인터페이스 또는 MemoryDB API를 사용하여 이벤트 메시지를 가져올 수 있습니다.

스냅샷 태깅

사용자 고유의 메타데이터를 태그 형태로 각 스냅샷에 할당할 수 있습니다. 태그를 사용하면 용도, 소유자 또는 환경을 기준으로 하는 등 스냅샷을 다양한 방식으로 분류할 수 있습니다. 이 기능은 동일 유형의 리소스가 많을 때 유용합니다. 지정한 태그에 따라 특정 리소스를 빠르게 식별할 수 있습니다. 자세한 내용은 태그 지정이 가능한 리소스 단원을 참조하십시오.

비용 할당 태그는 인보이스에 대한 비용을 태그 값으로 그룹화하여 여러 AWS서비스의 비용을 추적하는 수단입니다. 비용 할당 태그에 대해 자세히 알아보려면 비용 할당 태그 사용을 참조하세요.

MemoryDB 콘솔AWS CLI, 또는 MemoryDB API를 사용하여 스냅샷에 비용 할당 태그를 추가, 나열, 수정, 제거 또는 복사할 수 있습니다. 자세한 내용은 비용 할당 태그를 사용하여 비용 모니터링 단원을 참조하십시오.

스냅샷 삭제

보존 기간 제한이 만료되면 자동 스냅샷이 자동으로 삭제됩니다. 클러스터를 삭제하면 모든 자동 스냅샷도 삭제됩니다.

스냅샷이 자동으로 생성되건 수동으로 생성되건 관계없이 MemoryDB에서는 언제든지 스냅샷을 삭제할 수 있는 삭제 API 작업을 제공합니다. 수동 스냅샷에는 보존 제한이 없으므로 수동 삭제를 통해서만 수동 스냅샷을 제거할 수 있습니다.

MemoryDB 콘솔AWS CLI, 또는 MemoryDB API를 사용하여 스냅샷을 삭제할 수 있습니다.

스냅샷 삭제(콘솔)

다음 절차에서는 MemoryDB 콘솔을 사용하여 스냅샷을 삭제합니다.

스냅샷 삭제

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/memorydb/ MemoryDB 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Snapshots을 선택합니다.

   스냅샷 화면이 스냅샷 목록과 함께 나타납니다.
3. 삭제할 클러스터 이름 왼쪽에 있는 라디오 버튼을 선택합니다.
4. Actions를 선택하고 삭제를 선택합니다.
5. 이 스냅샷을 삭제하려면 텍스트 상자에 `delete`을 입력하고 삭제를 선택합니다. 삭제를 취소하려면 취소(cancel)를 선택합니다. 상태가 삭제 중(deleting)으로 변경됩니다.

스냅샷 삭제(AWSCLI)

다음 파라미터와 함께 delete-snapshot AWS CLI작업을 사용하여 스냅샷을 삭제합니다.

- `--snapshot-name` - 삭제할 스냅샷입니다.

다음 코드는 `myBackup` 스냅샷을 삭제합니다.

```
aws memorydb delete-snapshot --snapshot-name myBackup
```

자세한 내용은 AWS CLI 명령 참조의 delete-snapshot을 참조하세요.

스냅샷 삭제(MemoryDB API)

`DeleteSnapshot` API 작업을 다음 파라미터와 함께 사용하여 스냅샷을 삭제합니다.

- `SnapshotName` - 삭제할 스냅샷입니다.

다음 코드는 `myBackup` 스냅샷을 삭제합니다.

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DeleteSnapshot
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &SnapshotName=myBackup
   &Timestamp=20210802T192317Z
   &Version=2021-01-01
   &X-Amz-Credential=<credential>
```

자세한 내용은 DeleteSnapshot을 참조하세요.

스케일링

애플리케이션에서 처리해야 하는 데이터의 양은 거의 정적이 아닙니다. 비즈니스가 성장하거나 수요에서 일반적인 변동을 경험할 경우, 데이터의 양이 증가하거나 감소합니다. 애플리케이션을 자체적으로 관리할 경우, 최고의 수요에 대해 충분한 하드웨어를 프로비저닝해야 하므로, 비용이 많이 들 수 있습니다. MemoryDB를 사용하면 현재 수요에 맞게 규모를 조정하고, 사용한 만큼만 요금을 지불할 수 있습니다.

다음은 수행하려는 조정 작업에 대한 올바른 주제를 찾는 데 도움이 됩니다.

MemoryDB 스케일링

Action	MemoryDB	
확장	MemoryDB의 온라인 리샤딩	
노드 유형 변경	노드 유형 수정하여 온라인 수직 조정	
샤드 수 변경	MemoryDB 클러스터 크기 조정	

MemoryDB 클러스터 크기 조정

클러스터에 대한 수요 변화에 따라 MemoryDB 클러스터 내 샤드 수를 변경해 성능을 향상시키거나 비용을 줄이도록 결정할 수 있습니다. 이와 같이 하려면 온라인 수평적 조정을 사용하는 것이 좋은데, 이 방법은 조정 프로세스 중에도 클러스터가 계속해서 요청을 처리하도록 하기 때문입니다.

클러스터를 다시 조정하도록 결정할 수 있는 조건은 다음과 같습니다.

- 메모리 부족:

  클러스터의 노드에서 메모리가 부족하면 데이터를 저장 및 요청 처리에 더 많은 리소스를 사용하도록 확장을 결정할 수 있습니다.

  FreeableMemory, SwapUsage 및 BytesUsedForMemoryDB 지표를 모니터링해 노드에서 메모리가 부족한지 확인할 수 있습니다.
- CPU 또는 네트워크 병목 현상:

  클러스터에서 지연 시간/처리량 문제가 발생하면 문제를 해결하기 위해 확장이 필요할 수 있습니다.

  CPUUtilization, NetworkBytesIn, NetworkBytesOut, CurrConnections 및 NewConnections 지표를 모니터링해 지연 시간 및 처리량 수준을 모니터링할 수 있습니다.
- 클러스터가 과도하게 조정됨:

  축소와 같은 클러스터에 대한 현재 수요는 성능을 저하시키지 않고 비용을 줄입니다.

  다음 FreeableMemory, SwapUsage, BytesUseForCache, CPUUtilization, NetworkBytesIn, NetworkBytesOut, CurrConnections 및 NewConnections 지표를 사용하여 클러스터의 사용을 모니터링해 안전하게 스케일 인할 수 있는지 확인할 수 있습니다.

조정의 성능 영향

오프라인 프로세스를 사용해 조정하는 경우, 프로세스 중 상당 부분에서 클러스터가 오프라인 상태가 되기 때문에 요청을 처리할 수 없습니다. 온라인 방법을 사용해 조정하는 경우, 클러스터가 조정 작업 전체에서 계속해서 요청을 처리할 수 있음에도 불구하고 조정은 컴퓨팅 집약적인 작업이기 때문에 성능 저하가 발생합니다. 저하 정도는 일반적인 CPU 사용률과 데이터에 따라 달라집니다.

MemoryDB 클러스터를 조정하는 방법에는 수평 확장과 수직 확장이라는 두 가지 방법이 있습니다.

- 수평 조정에서는 샤드를 추가 또는 제거하여 클러스터 내 샤드 수를 변경할 수 있습니다. 온라인 리샤딩 프로세스를 통해 클러스터가 들어오는 요청을 계속 처리하는 동안 확장/축소할 수 있습니다.

- 수직 확장 - 노드 유형을 변경하여 클러스터의 크기를 조정합니다. 온라인 수직 확장을 통해 클러스터가 들어오는 요청을 계속 처리하는 동안 확장/축소할 수 있습니다.

스케일 인 또는 스케일 다운을 통해 클러스터의 크기와 메모리 용량을 줄이는 경우 새 구성에 데이터 및 엔진 오버헤드를 위한 충분한 메모리가 있는지 확인합니다.

MemoryDB의 오프라인 리샤딩

오프라인 리샤딩 재구성의 주요 이점은 클러스터에서 단순히 샤드를 추가 또는 제거하는 것 이상을 할 수 있다는 점입니다. 오프라인 리샤딩 시 클러스터의 샤드 수를 변경하는 것 이외에 다음을 수행할 수 있습니다.

- 클러스터의 노드 유형을 변경합니다.
- 최신 엔진 버전으로 업그레이드합니다.

> **Note**
> 데이터 계층화가 활성화된 클러스터에서는 오프라인 리샤딩이 지원되지 않습니다. 자세한 내용은 데이터 계층화 섹션을 참조하세요.

오프라인 샤드 재구성의 주요 단점은 프로세스의 복원 부분에서 클러스터가 오프라인 상태가 되어 애플리케이션에서 엔드포인트를 업데이트할 때까지 이 상태가 지속된다는 점입니다. 클러스터가 오프라인 상태도 지속되는 기간은 클러스터 내 데이터의 양에 따라 달라집니다.

샤드 MemoryDB 클러스터를 오프라인 상태에서 재구성하려면

1. 기존 MemoryDB 클러스터의 수동 스냅샷을 생성합니다. 자세한 내용은 수동 스냅샷 생성 섹션을 참조하세요.
2. 스냅샷에서 복원해 새 클러스터를 생성합니다. 자세한 내용은 스냅샷에서 복원 섹션을 참조하세요.
3. 애플리케이션에서 엔드포인트를 새 클러스터의 엔드포인트로 업데이트합니다. 자세한 내용은 연결 엔드포인트 찾기 섹션을 참조하세요.

MemoryDB의 온라인 리샤딩

MemoryDB에서 온라인 리샤딩을 사용하여 가동 중지 시간 없이 동적으로 MemoryDB의 규모를 조정할 수 있습니다. 이러한 접근 방식은 조정 또는 재분배 진행 중에도 클러스터에서 계속해서 요청을 처리할 수 있음을 의미합니다.

다음을 수행할 수 있습니다.

- 스케일 아웃 - MemoryDB 클러스터에 샤드를 추가하여 읽기 및 쓰기 용량을 늘립니다.

  클러스터에 샤드를 하나 이상 추가하는 경우, 각 샤드의 노드 수는 기존의 가장 작은 샤드에 있는 노드 수와 동일합니다.
- 스케일 인 – MemoryDB 클러스터에서 샤드를 제거해 읽기 및 쓰기 용량을 줄여 비용을 절감합니다.

현재, MemoryDB 온라인 리샤딩에는 다음 제한 사항이 적용됩니다.

- 슬롯 또는 키스페이스와 대용량 항목에 대한 제한 사항이 있습니다.

  샤드 내 키에 대용량 항목이 포함되어 있으면 스케일 아웃 시 해당 키가 새 샤드로 마이그레이션되지 않습니다. 이 기능으로 인해 불균형 샤드가 발생할 수 있습니다.

  샤드 내 키에 대용량 항목(직렬화 후 256MB보다 큰 항목)이 포함되어 있으면 축소 시 해당 샤드는 삭제되지 않습니다. 이 기능으로 인해 일부 샤드가 삭제되지 않을 수 있습니다.
- 스케일 아웃 시 새 샤드의 노드 수는 기존 샤드의 노드 수와 같습니다.

자세한 내용은 [모범 사례: 온라인 클러스터 크기 조정](#) 섹션을 참조하세요.

AWS Management 콘솔, AWS CLI 및 MemoryDB API를 사용하여 MemoryDB 클러스터를 수평적 스케일링할 수 있습니다.

온라인 리샤딩을 사용하여 샤드 추가

AWS Management 콘솔, AWS CLI 또는 MemoryDB API를 사용하여 MemoryDB 클러스터에 샤드를 추가할 수 있습니다.

샤드 추가(콘솔)

AWS Management 콘솔을 사용하여 MemoryDB 클러스터에 샤드를 하나 이상 추가할 수 있습니다. 다음 절차에서는 이러한 프로세스를 설명합니다.

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 클러스터 목록에서 샤드를 추가할 클러스터 이름을 선택합니다.
3. 샤드 및 노드 탭에서 샤드 추가/삭제를 선택합니다.
4. 새 샤드 수에 원하는 샤드 수를 입력합니다.
5. 변경 내용을 저장하려면 확인을 선택하고 취소하려면 취소를 선택합니다.

샤드 추가(AWS CLI)

다음 프로세스에서는 AWS CLI을(를) 사용해 샤드를 추가하여 MemoryDB 클러스터에서 샤드를 재구성하는 방법을 설명합니다.

`update-cluster`에 다음 파라미터를 사용합니다.

파라미터

- `--cluster-name` - 필수입니다. 샤드 재구성 작업을 수행할 클러스터를 지정합니다.
- `--shard-configuration` - 필수입니다. 샤드 수를 설정할 수 있습니다.
  - `ShardCount` - 이 속성을 설정하여 원하는 샤드 수를 지정합니다.

Example

다음 예제에서는 클러스터 `my-cluster`의 샤드 수를 2로 수정합니다.

Linux, macOS, Unix의 경우:

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --shard-configuration \
        ShardCount=2
```

Windows의 경우:

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --shard-configuration ^
        ShardCount=2
```

다음과 같은 JSON 응답을 반환합니다.


```
{
    "Cluster": {
        "Name": "my-cluster",
        "Status": "updating",
        "NumberOfShards": 2,
        "AvailabilityMode": "MultiAZ",
        "ClusterEndpoint": {
            "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-east-1.amazonaws.com",
            "Port": 6379
        },
        "NodeType": "db.r6g.large",
        "EngineVersion": "6.2",
        "EnginePatchVersion": "6.2.6",
        "ParameterGroupName": "default.memorydb-redis6",
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
        "SnapshotWindow": "04:30-05:30",
        "DataTiering": "false",
        "AutoMinorVersionUpgrade": true
    }
}
```


상태가 업데이트 중에서 사용 가능으로 변경된 후 업데이트된 클러스터의 세부 정보를 보려면 다음 명령을 사용하세요.

Linux, macOS, Unix의 경우:

```
aws memorydb describe-clusters \
    --cluster-name my-cluster
    --show-shard-details
```

Windows의 경우:

```
aws memorydb describe-clusters ^
    --cluster-name my-cluster
    --show-shard-details
```

다음과 같은 JSON 응답을 반환합니다.


```
{
    "Clusters": [
        {
            "Name": "my-cluster",
            "Status": "available",
            "NumberOfShards": 2,
            "Shards": [
                {
                    "Name": "0001",
                    "Status": "available",
                    "Slots": "0-8191",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0001-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0001-002",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1b",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        }
                    ],
                    "NumberOfNodes": 2
                },
                {
                    "Name": "0002",
```

```
                    "Status": "available",
                    "Slots": "8192-16383",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0002-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1b",
                            "CreateTime": "2021-08-22T14:26:18.693000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0002-002",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-22T14:26:18.765000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        }
                    ],
                    "NumberOfNodes": 2
                }
            ],
            "ClusterEndpoint": {
                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
            },
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "my-sg",
            "TLSEnabled": true,
            "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
            "MaintenanceWindow": "wed:03:00-wed:04:00",
```

```
            "SnapshotWindow": "04:30-05:30",
            "ACLName": "my-acl",
            "DataTiering": "false",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```

자세한 내용은 AWS CLI 명령 레퍼런스의 update-cluster을(를) 참조하세요.

샤드 추가(MemoryDB API)

MemoryDB API에서 `UpdateCluster` 작업을 사용해 MemoryDB 클러스터 내 샤드를 온라인으로 재구성할 수 있습니다.

`UpdateCluster`에 다음 파라미터를 사용합니다.

파라미터

- `ClusterName` - 필수입니다. 샤드 재구성 작업을 수행할 클러스터를 지정합니다.
- `ShardConfiguration` - 필수입니다. 샤드 수를 설정할 수 있습니다.
  - `ShardCount` - 이 속성을 설정하여 원하는 샤드 수를 지정합니다.

자세한 내용은 UpdateCluster를 참조하세요.

온라인 리샤딩을 사용하여 샤드 제거

AWS Management 콘솔, AWS CLI 또는 MemoryDB API를 사용하여 클러스터에서 샤드를 제거할 수 있습니다.

샤드 제거(콘솔)

다음 프로세스에서는 AWS Management 콘솔을(를) 사용해 샤드를 제거하여 MemoryDB 클러스터에서 샤드를 재구성하는 방법을 설명합니다.

> **Important**
>
> 클러스터에서 샤드를 제거하기 전에 MemoryDB에서는 모든 데이터가 나머지 샤드에 맞는지 확인합니다. 데이터가 맞으면 요청에 따라 샤드가 클러스터에서 삭제됩니다. 데이터가 나머지 샤드에 맞지 않으면 프로세스가 종료되고 클러스터는 요청이 작성되기 전과 동일한 샤드 구성으로 남습니다.

AWS Management 콘솔을(를) 사용하여 MemoryDB 클러스터에서 샤드를 하나 이상 제거할 수 있습니다. 클러스터의 모든 샤드를 제거할 수는 없습니다. 대신 클러스터를 삭제해야 합니다. 자세한 내용은 5단계: 클러스터 삭제 섹션을 참조하세요. 다음 절차는 샤드를 하나 이상 삭제하는 프로세스를 설명합니다.

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 클러스터 목록에서 샤드를 제거할 클러스터 이름을 선택합니다.
3. 샤드 및 노드 탭에서 샤드 추가/삭제를 선택합니다.
4. 새 샤드 수에 원하는 샤드 수(최소 1개)를 입력합니다.
5. 변경 내용을 저장하려면 확인을 선택하고, 취소하려면 취소를선택합니다.

샤드 제거(AWS CLI)

다음 프로세스에서는 AWS CLI을(를) 사용해 샤드를 제거하여 MemoryDB 클러스터에서 샤드를 재구성하는 방법을 설명합니다.

> **Important**
>
> 클러스터에서 샤드를 제거하기 전에 MemoryDB에서는 모든 데이터가 나머지 샤드에 맞는지 확인합니다. 데이터가 맞으면 요청된 대로 샤드가 클러스터에서 삭제되고 해당 샤드의 키스페이스가 나머지 샤드로 매핑됩니다. 데이터가 나머지 샤드에 맞지 않으면 프로세스가 종료되고 클러스터는 요청이 작성되기 전과 동일한 샤드 구성으로 남습니다.

AWS CLI을(를) 사용하여 MemoryDB 클러스터에서 샤드를 하나 이상 제거할 수 있습니다. 클러스터의 모든 샤드를 제거할 수는 없습니다. 대신 클러스터를 삭제해야 합니다. 자세한 내용은 5단계: 클러스터 삭제 섹션을 참조하세요.

`update-cluster`에 다음 파라미터를 사용합니다.

파라미터

- `--cluster-name` - 필수입니다. 샤드 재구성 작업을 수행할 클러스터를 지정합니다.
- `--shard-configuration` - 필수입니다. `ShardCount` 속성을 사용하여 샤드 수를 설정할 수 있습니다.

ShardCount - 이 속성을 설정하여 원하는 샤드 수를 지정합니다.

Example

다음 예제에서는 클러스터 my-cluster의 샤드 수를 2로 수정합니다.

Linux, macOS, Unix의 경우:

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --shard-configuration \
        ShardCount=2
```

Windows의 경우:

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --shard-configuration ^
        ShardCount=2
```

다음과 같은 JSON 응답을 반환합니다.

```
{
    "Cluster": {
        "Name": "my-cluster",
        "Status": "updating",
        "NumberOfShards": 2,
        "AvailabilityMode": "MultiAZ",
        "ClusterEndpoint": {
            "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-east-1.amazonaws.com",
            "Port": 6379
        },
        "NodeType": "db.r6g.large",
        "EngineVersion": "6.2",
        "EnginePatchVersion": "6.2.6",
        "ParameterGroupName": "default.memorydb-redis6",
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
        "SnapshotRetentionLimit": 0,
```

```
            "MaintenanceWindow": "wed:03:00-wed:04:00",
            "SnapshotWindow": "04:30-05:30",
            "DataTiering": "false",
            "AutoMinorVersionUpgrade": true
        }
}
```

상태가 업데이트 중에서 사용 가능으로 변경된 후 업데이트된 클러스터의 세부 정보를 보려면 다음 명령을 사용하세요.

Linux, macOS, Unix의 경우:

```
aws memorydb describe-clusters \
    --cluster-name my-cluster
    --show-shard-details
```

Windows의 경우:

```
aws memorydb describe-clusters ^
    --cluster-name my-cluster
    --show-shard-details
```

다음과 같은 JSON 응답을 반환합니다.

```
{
    "Clusters": [
        {
            "Name": "my-cluster",
            "Status": "available",
            "NumberOfShards": 2,
            "Shards": [
                {
                    "Name": "0001",
                    "Status": "available",
                    "Slots": "0-8191",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0001-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
```

```
                        "Endpoint": {
                            "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                            "Port": 6379
                        }
                    },
                    {
                        "Name": "my-cluster-0001-002",
                        "Status": "available",
                        "AvailabilityZone": "us-east-1b",
                        "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                        "Endpoint": {
                            "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                            "Port": 6379
                        }
                    }
                ],
                "NumberOfNodes": 2
            },
            {
                "Name": "0002",
                "Status": "available",
                "Slots": "8192-16383",
                "Nodes": [
                    {
                        "Name": "my-cluster-0002-001",
                        "Status": "available",
                        "AvailabilityZone": "us-east-1b",
                        "CreateTime": "2021-08-22T14:26:18.693000-07:00",
                        "Endpoint": {
                            "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                            "Port": 6379
                        }
                    },
                    {
                        "Name": "my-cluster-0002-002",
                        "Status": "available",
                        "AvailabilityZone": "us-east-1a",
                        "CreateTime": "2021-08-22T14:26:18.765000-07:00",
                        "Endpoint": {
                            "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
```

```
                                "Port": 6379
                            }
                        }
                    ],
                    "NumberOfNodes": 2
                }
            ],
            "ClusterEndpoint": {
                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
            },
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "my-sg",
            "TLSEnabled": true,
            "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
            "MaintenanceWindow": "wed:03:00-wed:04:00",
            "SnapshotWindow": "04:30-05:30",
            "ACLName": "my-acl",
            "DataTiering": "false",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```

자세한 내용은 AWS CLI 명령 레퍼런스의 update-cluster를 참조하세요.

샤드 제거(MemoryDB API)

MemoryDB API에서 `UpdateCluster` 작업을 사용해 MemoryDB 클러스터 내 샤드를 온라인으로 재구성할 수 있습니다.

다음 프로세스에서는 MemoryDB API를 사용해 샤드를 제거하여 MemoryDB 클러스터에서 샤드를 재구성하는 방법을 설명합니다.

> **Important**
>
> 클러스터에서 샤드를 제거하기 전에 MemoryDB에서는 모든 데이터가 나머지 샤드에 맞는지 확인합니다. 데이터가 맞으면 요청된 대로 샤드가 클러스터에서 삭제되고 해당 샤드의 키스페이스가 나머지 샤드로 매핑됩니다. 데이터가 나머지 샤드에 맞지 않으면 프로세스가 종료되고 클러스터는 요청이 작성되기 전과 동일한 샤드 구성으로 남습니다.

MemoryDB API를 사용하여 MemoryDB 클러스터에서 샤드를 하나 이상 제거할 수 있습니다. 클러스터의 모든 샤드를 제거할 수는 없습니다. 대신 클러스터를 삭제해야 합니다. 자세한 내용은 [5단계: 클러스터 삭제](#) 섹션을 참조하세요.

`UpdateCluster`에 다음 파라미터를 사용합니다.

파라미터

- `ClusterName` - 필수입니다. 샤드 재구성 작업을 수행할 클러스터를 지정합니다.
- `ShardConfiguration` - 필수입니다. `ShardCount` 속성을 사용하여 샤드 수를 설정할 수 있습니다.

  `ShardCount` - 이 속성을 설정하여 원하는 샤드 수를 지정합니다.

노드 유형 수정하여 온라인 수직 조정

MemoryDB로 온라인 수직 조정을 사용하여 중단 시간 없이 동적으로 클러스터를 조정할 수 있습니다. 이를 통해 클러스터는 조정 중에도 요청을 처리할 수 있습니다.

> **Note**
>
> 데이터 계층화 클러스터(예: r6gd 노드 유형을 사용하는 클러스터)와 데이터 계층화를 사용하지 않는 클러스터(예: r6g 노드 유형을 사용하는 클러스터) 간에는 크기 조정이 지원되지 않습니다. 자세한 내용은 [데이터 계층화](#) 섹션을 참조하세요.

다음을 수행할 수 있습니다.

- 스케일 업 - MemoryDB 클러스터의 노드 유형을 더 큰 노드 유형을 사용하도록 조정하여 읽기 및 쓰기 용량을 늘립니다.

MemoryDB에서는 온라인 상태로 요청을 처리하는 동안 클러스터 크기를 동적으로 조정합니다.

- 축소 - 더 작은 노드를 사용하도록 노드 유형을 조정하여 읽기 및 쓰기 용량을 줄입니다. 마찬가지로, MemoryDB에서는 온라인 상태로 요청을 처리하는 동안 클러스터 크기를 동적으로 조정합니다. 이 경우, 노드를 축소하여 비용을 절감할 수 있습니다.

> **Note**
>
> 확장 및 축소 프로세스는 새로 선택한 노드 유형을 사용하여 클러스터를 생성하고 새 노드를 이전 노드와 동기화합니다. 원활한 확장/축소 흐름을 위해 다음을 수행합니다.
>
> - 수직 확장 프로세스는 온라인 상태를 유지하도록 설계되었지만 이전 노드와 새 노드 간의 데이터 동기화에 의존합니다. 데이터 트래픽이 최소 수준일 것으로 예상되는 시간 동안 확장/축소를 시작하는 것이 좋습니다.
> - 가능한 경우, 준비 환경에서 조정 중 애플리케이션 동작을 테스트합니다.

온라인 확장

주제

- [MemoryDB 클러스터 스케일 업(콘솔)]
- [MemoryDB 클러스터 스케일업(AWS CLI)]
- [MemoryDB 클러스터 스케일 업(MemoryDB API)]

MemoryDB 클러스터 스케일 업(콘솔)

다음 절차에서는 AWS Management 콘솔을(를) 사용하여 MemoryDB 클러스터를 스케일 업하는 방법에 대해 설명합니다. 이 프로세스 동안 MemoryDB 클러스터는 가동 중지 시간을 최소화하면서 요청을 계속 처리합니다.

클러스터를 스케일 업하려면(콘솔)

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 클러스터 목록에서 클러스터를 선택합니다.
3. 작업을 선택한 다음 수정을 선택합니다.

4. 클러스터 수정 대화 상자에서:

   • [Node type] 목록에서 조정할 노드 유형을 선택합니다. 확장하려면, 기존 노드보다 큰 노드 유형을 선택합니다.

5. 변경 사항 저장을 선택합니다.

   클러스터의 상태가 수정 중으로 변경됩니다. 상태가 사용 가능으로 변경되면 수정이 완료되고 새 클러스터의 사용을 시작할 수 있습니다.

MemoryDB 클러스터 스케일업(AWS CLI)

다음 절차에서는 AWS CLI을(를) 사용하여 MemoryDB 클러스터를 스케일 업하는 방법에 대해 설명합니다. 이 프로세스 동안 MemoryDB 클러스터는 가동 중지 시간을 최소화하면서 요청을 계속 처리합니다.

MemoryDB 클러스터를 스케일 업하려면(AWS CLI)

1. 다음 파라미터와 함께 AWS CLI`list-allowed-node-type-updates`명령을 실행하여 확장할 수 있는 노드 유형을 확인합니다.

   Linux, macOS, Unix의 경우:

   ```
   aws memorydb list-allowed-node-type-updates \
        --cluster-name my-cluster-name
   ```

   Windows의 경우:

   ```
   aws memorydb list-allowed-node-type-updates ^
        --cluster-name my-cluster-name
   ```

   위 명령의 출력은 다음과 같습니다(JSON 형식).

   ```
   {
        "ScaleUpNodeTypes": [
            "db.r6g.2xlarge",
            "db.r6g.large"
        ],
        "ScaleDownNodeTypes": [
            "db.r6g.large"
        ],
   ```

```
}
```

자세한 내용은 AWS CLI참조의 list-allowed-node-type-updates을(를) 참조하세요.

2. 다음 파라미터와 함께 AWS CLI `update-cluster` 명령을 사용하여 클러스터를 수정하여 새롭고 더 큰 노드 유형으로 스케일 업합니다.

   - `--cluster-name` - 스케일 업할 클러스터의 이름입니다.
   - `--node-type` - 클러스터를 조정할 새 노드 유형입니다. 이 값은 1단계의 `list-allowed-node-type-updates` 명령에 의해 반환되는 노드 유형 중 하나여야 합니다.

   Linux, macOS, Unix의 경우:

   ```
   aws memorydb update-cluster  \
    --cluster-name my-cluster \
    --node-type db.r6g.2xlarge
   ```

   Windows의 경우:

   ```
   aws memorydb update-cluster ^
        --cluster-name my-cluster ^
        --node-type db.r6g.2xlarge ^
   ```

   자세한 내용은 update-cluster 단원을 참조하세요.

MemoryDB 클러스터 스케일 업(MemoryDB API)

다음 절차는 MemoryDB API를 사용하여 캐시 클러스터를 현재 노드 유형에서 새롭고 더 큰 노드 유형으로 조정합니다. 이 프로세스 중에 MemoryDB는 DNS 항목이 새 노드를 가리키도록 해당 항목을 업데이트합니다. 클러스터가 온라인 상태에서 들어오는 요청을 계속 처리하는 동안 자동 장애 조치가 활성화된 클러스터를 조정할 수 있습니다.

대형 노드 유형으로 스케일 업하는 데 걸리는 시간은 노드 유형 및 현재 클러스터에 있는 데이터의 양에 따라 달라집니다.

MemoryDB 클러스터를 스케일 업하려면(MemoryDB API)

1. 다음 파라미터와 함께 MemoryDB API `ListAllowedNodeTypeUpdates` 작업을 사용하여 스케일 업할 수 있는 노드 유형을 확인합니다.

   - `ClusterName` – 클러스터의 이름입니다. 모든 클러스터 대신 특정 클러스터를 설명하려면 이 파라미터를 사용하세요.

   ```
   https://memory-db.us-east-1.amazonaws.com/
       ?Action=ListAllowedNodeTypeUpdates
       &ClusterName=MyCluster
       &Version=2021-01-01
       &SignatureVersion=4
       &SignatureMethod=HmacSHA256
       &Timestamp=20210802T192317Z
       &X-Amz-Credential=<credential>
   ```

   자세한 내용은 MemoryDB API 참조의 ListAllowedNodeTypeUpdates를 참조하세요.

2. 다음 파라미터와 함께 `UpdateCluster` MemoryDB API 작업을 사용하여 현재 클러스터를 새 노드 유형으로 확장합니다.

   - `ClusterName` – 클러스터의 이름입니다.
   - `NodeType`- 이 클러스터에 있는 클러스터의 새롭고 더 큰 노드 유형입니다. 이 값은 1단계의 `ListAllowedNodeTypeUpdates` 작업에 의해 반환되는 인스턴스 유형 중 하나여야 합니다.

   ```
   https://memory-db.us-east-1.amazonaws.com/
       ?Action=UpdateCluster
       &NodeType=db.r6g.2xlarge
       &ClusterName=myCluster
       &SignatureVersion=4
       &SignatureMethod=HmacSHA256
       &Timestamp=20210801T220302Z
       &Version=2021-01-01
       &X-Amz-Algorithm=Amazon4-HMAC-SHA256
       &X-Amz-Date=20210801T220302Z
       &X-Amz-SignedHeaders=Host
       &X-Amz-Expires=20210801T220302Z
       &X-Amz-Credential=<credential>
       &X-Amz-Signature=<signature>
   ```

자세한 내용은 UpdateCluster를 참조하세요.

온라인 축소

주제

- MemoryDB 클러스터 축소(콘솔)
- MemoryDB 클러스터 스케일 다운(AWS CLI)
- MemoryDB 클러스터 규모 축소(MemoryDB API)

MemoryDB 클러스터 축소(콘솔)

다음 절차에서는 AWS Management 콘솔을(를) 사용하여 MemoryDB 클러스터를 축소하는 방법에 대해 설명합니다. 이 프로세스 동안 MemoryDB 클러스터는 가동 중지 시간을 최소화하면서 요청을 계속 처리합니다.

MemoryDB 클러스터를 축소하려면(콘솔)

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 클러스터 목록에서 원하는 클러스터를 선택합니다.
3. 작업을 선택한 다음 수정을 선택합니다.
4. 클러스터 수정 대화 상자에서:
   - [Node type] 목록에서 조정할 노드 유형을 선택합니다. 축소하려면, 기존 노드보다 작은 노드 유형을 선택합니다. 모든 노드 유형을 축소할 수 있는 것은 아닙니다.
5. 변경 사항 저장을 선택합니다.

   클러스터의 상태가 수정 중으로 변경됩니다. 상태가 사용 가능으로 변경되면 수정이 완료되고 새 클러스터의 사용을 시작할 수 있습니다.

MemoryDB 클러스터 스케일 다운(AWS CLI)

다음 절차에서는 AWS CLI를 사용하여 MemoryDB 클러스터를 축소하는 방법에 대해 설명합니다. 이 프로세스 동안 MemoryDB 클러스터는 가동 중지 시간을 최소화하면서 요청을 계속 처리합니다.

MemoryDB 클러스터 축소(CLI AWS)

1. 다음 파라미터와 함께 AWS CLI `list-allowed-node-type-updates` 명령을 실행하여 축소할 수 있는 노드 유형을 확인합니다.

   Linux, macOS, Unix의 경우:

   ```
   aws memorydb list-allowed-node-type-updates \
        --cluster-name my-cluster-name
   ```

   Windows의 경우:

   ```
   aws memorydb list-allowed-node-type-updates ^
        --cluster-name my-cluster-name
   ```

   위 명령의 출력은 다음과 같습니다(JSON 형식).

   ```
   {
        "ScaleUpNodeTypes": [
            "db.r6g.2xlarge",
            "db.r6g.large"
        ],
        "ScaleDownNodeTypes": [
            "db.r6g.large"
        ],
   }
   ```

   자세한 내용은 list-allowed-node-type-updates를 참조하세요.

2. 다음 파라미터와 함께 `update-cluster` 명령을 사용하여 클러스터를 수정하여 새롭고 더 작은 노드 유형으로 축소합니다.

   - `--cluster-name` - 축소할 캐시 클러스터의 이름입니다.
   - `--node-type` - 클러스터를 조정할 새 노드 유형입니다. 이 값은 1단계의 `list-allowed-node-type-updates` 명령에 의해 반환되는 노드 유형 중 하나여야 합니다.

   Linux, macOS, Unix의 경우:

   ```
   aws memorydb update-cluster  \
        --cluster-name my-cluster \
   ```

```
    --node-type db.r6g.large
```

Windows의 경우:

```
aws memorydb update-cluster ^
     --cluster-name my-cluster ^
     --node-type db.r6g.large
```

자세한 내용은 update-cluster 단원을 참조하세요.

MemoryDB 클러스터 규모 축소(MemoryDB API)

다음 절차는 MemoryDBAPI를 사용하여 클러스터를 현재 노드 유형에서 새롭고 더 작은 노드 유형으로 조정합니다. 이 프로세스 동안 MemoryDB 클러스터는 가동 중지 시간을 최소화하면서 요청을 계속 처리합니다.

더 작은 노드 유형으로 축소하는 데 걸리는 시간은 노드 유형 및 현재 클러스터에 있는 데이터의 양에 따라 달라집니다.

축소 (MemoryDB API)

1. 다음 파라미터와 함께 ListAllowedNodeTypeUpdates API 작업을 사용하여 축소할 수 있는 노드 유형을 확인합니다.

   - `ClusterName` – 클러스터의 이름입니다. 모든 클러스터 대신 특정 클러스터를 설명하려면 이 파라미터를 사용하세요.

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=ListAllowedNodeTypeUpdates
    &ClusterName=MyCluster
    &Version=2021-01-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210802T192317Z
    &X-Amz-Credential=<credential>
```

2. 다음 파라미터와 함께 UpdateCluster API 작업을 사용하여 현재 클러스터를 새 노드 유형으로 축소합니다.

   - `ClusterName` – 클러스터의 이름입니다.

- NodeType- 이 클러스터에 있는 클러스터의 새롭고 더 작은 노드 유형입니다. 이 값은 1단계의 ListAllowedNodeTypeUpdates 작업에 의해 반환되는 인스턴스 유형 중 하나여야 합니다.

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=UpdateCluster
    &NodeType=db.r6g.2xlarge
    &ClusterName=myReplGroup
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

파라미터 그룹을 사용해 엔진 파라미터 구성

MemoryDB는 파라미터를 사용하여 노드 및 클러스터의 런타임 속성을 제어합니다. 일반적으로 최신 엔진 버전에는 새로운 기능을 기원하는 추가 파라미터가 포함됩니다. 파라미터가 정리된 표는 엔진별 파라미터 섹션을 참조하세요.

maxmemory와 같은 일부 파라미터 값은 엔진 및 노드 유형에 의해 결정됩니다. 노드 유형별 파라미터 값의 표는 MemoryDB 노드 유형별 파라미터 섹션을 참조하세요.

주제

파라미터 관리

더욱 쉬운 파라미터 관리를 위해 파라미터를 명명된 파라미터 그룹으로 그룹화합니다. 파라미터 그룹은 시작하는 동안 엔진 소프트웨어에 전달되는 파라미터의 특정 값 조합을 나타냅니다. 이 값은 각 노드의 엔진 프로세서가 런타임에 작동하는 방식을 결정합니다. 특정 파라미터 그룹의 파라미터 값은 해당 파라미터가 속한 클러스터와 상관없이 그룹과 연결된 모든 노드에 적용됩니다.

클러스터 성능을 미세 조정하려면 일부 파라미터 값을 수정하거나 클러스터의 파라미터 그룹을 변경할 수 있습니다.

- 기본 파라미터 그룹을 수정하거나 삭제할 수 없습니다. 사용자 지정 파라미터 값이 필요하면 사용자 지정 파라미터 그룹을 생성해야 합니다.
- 파라미터 그룹 패밀리와 할당할 클러스터는 호환 가능해야 합니다. 예를 들어 클러스터에서 Redis OSS 버전 6을 실행 중이라면 memorydb_redis6 패밀리의 기본 또는 사용자 지정 파라미터 그룹만 사용할 수 있습니다.
- 클러스터의 파라미터를 변경하면 변경 사항이 클러스터에 즉시 적용됩니다. 이는 클러스터의 파라미터 그룹 자체에서 변경하든 파라미터 값을 클러스터의 파라미터 그룹 내에서 변경하든 마찬가지입니다.

파라미터 그룹 티어

MemoryDB 파라미터 그룹 계층

전역 기본값

해당 리전의 모든 MemoryDB 고객을 위한 최상위 루트 파라미터 그룹입니다.

전역 기본 파라미터 그룹:

- MemoryDB용으로 예약되어 있으며 고객이 사용할 수 없습니다.

고객 기본값

고객의 사용을 위해 생성된 전역 기본 파라미터 그룹의 사본입니다.

고객 기본 파라미터 그룹:

- MemoryDB가 생성하고 소유합니다.
- 고객이 이 파라미터 그룹에서 지원하는 엔진 버전을 실행 중인 모든 클러스터의 파라미터 그룹으로 사용할 수 있습니다.
- 고객이 편집할 수 없습니다.

고객 소유

고객 기본 파라미터 그룹의 사본입니다. 고객 소유 파라미터 그룹은 고객이 파라미터 그룹을 생성할 때마다 만들어집니다.

고객 소유 파라미터 그룹:

- 고객이 생성하고 소유합니다.
- 고객의 호환 가능한 모든 클러스터에 할당할 수 있습니다.
- 고객이 사용자 지정 파라미터 그룹을 생성하기 위해 수정할 수 있습니다.

  모든 파라미터 값을 수정할 수 있는 것은 아닙니다. 자세한 내용은 엔진별 파라미터 단원을 참조하십시오.

파라미터 그룹 생성

기본값에서 변경하려는 파라미터 값이 하나 이상이면 새 파라미터 그룹을 생성해야 합니다. MemoryDB 콘솔AWS CLI, 또는 MemoryDB API를 사용하여 파라미터 그룹을 생성할 수 있습니다.

파라미터 그룹 생성(콘솔)

다음 절차는 MemoryDB 콘솔을 사용하여 파라미터 그룹을 생성하는 방법을 보여줍니다.

MemoryDB 콘솔을 사용하여 파라미터 그룹을 생성하려면

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/memorydb/ MemoryDB 콘솔을 엽니다.
2. 사용 가능한 모든 파라미터 목록을 표시하려면 왼쪽 탐색 창에서 [Parameter Groups]를 선택합니다.
3. 파라미터 그룹을 생성하려면 Create parameter group을 선택합니다.

   Create parameter group 페이지가 나타납니다.
4. [Name] 상자에 파라미터 그룹의 고유 이름을 입력합니다.

   클러스터를 생성하거나 클러스터의 파라미터 그룹을 수정할 때 그 이름으로 파라미터 그룹을 선택합니다. 그러므로 이름은 파라미터 그룹의 패밀리를 식별할 수 있고 정보를 알 수 있는 것이 좋습니다.

   파라미터 그룹 명명 제약 조건은 다음과 같습니다.

   - ASCII 문자로 시작해야 합니다.
   - ASCII 문자, 숫자 및 하이픈만 포함할 수 있습니다.
   - 1-255자여야 합니다.
   - 하이픈 2개가 연속될 수 없습니다.
   - 끝에 하이픈이 올 수 없습니다.
5. [Description] 상자에 파라미터 그룹에 대한 설명을 입력합니다.
6. 엔진 버전 호환성 상자에서 이 파라미터 그룹에 해당하는 엔진 버전을 선택합니다.
7. 태그에서 선택적으로 태그를 추가하여 파라미터 그룹을 검색 및 필터링하거나 AWS비용을 추적합니다.
8. 파라미터 그룹을 생성하려면 [Create]를 선택합니다.

파라미터 그룹을 생성하지 않고 프로세스를 종료하려면 [Cancel]을 선택합니다.

9. 파라미터 그룹을 생성하면 패밀리의 기본값이 부여됩니다. 기본값을 변경하려면 파라미터 그룹을 수정해야 합니다. 자세한 내용은 [파라미터 그룹 수정](#) 단원을 참조하십시오.

파라미터 그룹 생성(AWSCLI)

를 사용하여 파라미터 그룹을 생성하려면 다음 파라미터`create-parameter-group`와 함께 명령을 AWS CLI사용합니다.

- `--parameter-group-name` - 파라미터 그룹의 이름입니다.

  파라미터 그룹 명명 제약 조건은 다음과 같습니다.
  - ASCII 문자로 시작해야 합니다.
  - ASCII 문자, 숫자 및 하이픈만 포함할 수 있습니다.
  - 1-255자여야 합니다.
  - 하이픈 2개가 연속될 수 없습니다.
  - 끝에 하이픈이 올 수 없습니다.
- `--family` - 파라미터 그룹의 엔진 및 버전 패밀리입니다.
- `--description` - 사용자가 정의한 파라미터 그룹에 대한 설명입니다.

Example

다음 예제에서는 memorydb_redis6 패밀리를 템플릿으로 사용하여 myRedis6x이라는 파라미터 그룹을 생성합니다.

Linux, macOS, Unix의 경우:

```
aws memorydb create-parameter-group \
    --parameter-group-name myRedis6x  \
    --family memorydb_redis6 \
    --description "My first parameter group"
```

Windows의 경우:

```
aws memorydb create-parameter-group ^
    --parameter-group-name myRedis6x  ^
```

```
    --family memorydb_redis6 ^
    --description "My first parameter group"
```

이 명령의 출력은 다음과 유사해야 합니다.

```
{
    "ParameterGroup": {
        "Name": "myRedis6x",
        "Family": "memorydb_redis6",
        "Description": "My first parameter group",
        "ARN": "arn:aws:memorydb:us-east-1:012345678912:parametergroup/myredis6x"
    }
}
```

파라미터 그룹을 생성하면 패밀리의 기본값이 부여됩니다. 기본값을 변경하려면 파라미터 그룹을 수정해야 합니다. 자세한 내용은 파라미터 그룹 수정 단원을 참조하십시오.

자세한 내용은 `create-parameter-group` 단원을 참조하십시오.

파라미터 그룹 생성(MemoryDB API)

MemoryDB API를 사용하여 파라미터 그룹을 생성하려면 이 파라미터와 함께 `CreateParameterGroup` 작업을 사용합니다.

- `ParameterGroupName` - 파라미터 그룹의 이름입니다.

  파라미터 그룹 명명 제약 조건은 다음과 같습니다.
  - ASCII 문자로 시작해야 합니다.
  - ASCII 문자, 숫자 및 하이픈만 포함할 수 있습니다.
  - 1-255자여야 합니다.
  - 하이픈 2개가 연속될 수 없습니다.
  - 끝에 하이픈이 올 수 없습니다.
- `Family` - 파라미터 그룹의 엔진 및 버전 패밀리입니다. 예를 들어 `memorydb_redis6`입니다.
- `Description` - 사용자가 정의한 파라미터 그룹에 대한 설명입니다.

Example

다음 예제에서는 memorydb_redis6 패밀리를 템플릿으로 사용하여 myRedis6x이라는 파라미터 그룹을 생성합니다.

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=CreateParameterGroup
    &Family=memorydb_redis6
    &ParameterGroupName=myRedis6x
    &Description=My%20first%20parameter%20group
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210802T192317Z
    &Version=2021-01-01
    &X-Amz-Credential=<credential>
```

이 작업의 응답은 다음과 유사해야 합니다.

```
<CreateParameterGroupResponse xmlns="http://memory-db.us-east-1.amazonaws.com/
doc/2021-01-01/">
  <CreateParameterGroupResult>
    <ParameterGroup>
      <Name>myRedis6x</Name>
      <Family>memorydb_redis6</Family>
      <Description>My first parameter group</Description>
      <ARN>arn:aws:memorydb:us-east-1:012345678912:parametergroup/myredis6x</ARN>
    </ParameterGroup>
  </CreateParameterGroupResult>
  <ResponseMetadata>
    <RequestId>d8465952-af48-11e0-8d36-859edca6f4b8</RequestId>
  </ResponseMetadata>
</CreateParameterGroupResponse>
```

파라미터 그룹을 생성하면 패밀리의 기본값이 부여됩니다. 기본값을 변경하려면 파라미터 그룹을 수정해야 합니다. 자세한 내용은 파라미터 그룹 수정 단원을 참조하십시오.

자세한 내용은 `CreateParameterGroup` 단원을 참조하십시오.

이름별로 파라미터 그룹 목록 조회

MemoryDB 콘솔, AWS CLI또는 MemoryDB API를 사용하여 파라미터 그룹을 나열할 수 있습니다.

이름별로 파라미터 그룹 목록 조회(콘솔)

다음 절차에서는 MemoryDB 콘솔을 사용하여 파라미터 그룹 목록을 보는 방법을 설명합니다.

MemoryDB 콘솔을 사용하여 파라미터 그룹을 나열하려면

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/memorydb/ MemoryDB 콘솔을 엽니다.
2. 사용 가능한 모든 파라미터 목록을 표시하려면 왼쪽 탐색 창에서 [Parameter Groups]를 선택합니다.

이름별 파라미터 그룹 나열(AWSCLI)

를 사용하여 파라미터 그룹 목록을 생성하려면 명령을 AWS CLI사용합니다`describe-parameter-groups`. 파라미터 그룹의 이름을 입력하면 그 파라미터 그룹만 나열됩니다. 파라미터 그룹의 이름을 입력하지 않으면 최대 `--max-results`개의 파라미터 그룹이 나열됩니다. 두 경우 모두 파라미터 그룹의 이름, 패밀리 및 설명이 나열됩니다.

Example

다음 샘플 코드에는 myRedis6x 파라미터 그룹이 나열되어 있습니다.

Linux, macOS, Unix의 경우:

```
aws memorydb describe-parameter-groups \
    --parameter-group-name myRedis6x
```

Windows의 경우:

```
aws memorydb describe-parameter-groups ^
    --parameter-group-name myRedis6x
```

이 명령의 출력은 파라미터 그룹의 이름, 패밀리 및 설명을 나열하는 것과 같습니다.

```
{
    "ParameterGroups": [
```

```
    {
        "Name": "myRedis6x",
        "Family": "memorydb_redis6",
        "Description": "My first parameter group",
               "ARN": "arn:aws:memorydb:us-east-1:012345678912:parametergroup/
myredis6x"
    }
   ]
}
```

Example

다음 샘플 코드에는 Valkey 또는 Redis OSS 엔진 버전 5.0.6 이상에서 실행되는 파라미터 그룹에 대한 파라미터 그룹 myRedis6x가 나열됩니다.

Linux, macOS, Unix의 경우:

```
aws memorydb describe-parameter-groups \
    --parameter-group-name myRedis6x
```

Windows의 경우:

```
aws memorydb describe-parameter-groups ^
    --parameter-group-name myRedis6x
```

이 명령의 출력은 파라미터 그룹의 이름, 패밀리 및 설명을 나열하는 것과 같습니다.

```
{
    "ParameterGroups": [
     {
        "Name": "myRedis6x",
        "Family": "memorydb_redis6",
        "Description": "My first parameter group",
               "ARN": "arn:aws:memorydb:us-east-1:012345678912:parametergroup/
myredis6x"
     }
    ]
}
```

Example

다음 샘플 코드는 최대 20개의 파라미터 그룹을 나열합니다.

```
aws memorydb describe-parameter-groups --max-results 20
```

이 명령의 JSON 출력은 각 파라미터 그룹의 이름, 패밀리 및 설명을 나열하는 것과 같습니다.

```
{
    "ParameterGroups": [
        {
            "ParameterGroupName": "default.memorydb-redis6",
            "Family": "memorydb_redis6",
            "Description": "Default parameter group for memorydb_redis6",
            "ARN": "arn:aws:memorydb:us-east-1:012345678912:parametergroup/
default.memorydb-redis6"
        },
        ...
    ]
}
```

자세한 내용은 describe-parameter-groups 단원을 참조하십시오.

이름으로 파라미터 그룹 나열(MemoryDB API)

MemoryDB API를 사용하여 파라미터 그룹의 목록을 생성하려면 `DescribeParameterGroups` 작업을 사용합니다. 파라미터 그룹의 이름을 입력하면 그 파라미터 그룹만 나열됩니다. 파라미터 그룹의 이름을 입력하지 않으면 최대 `MaxResults`개의 파라미터 그룹이 나열됩니다. 두 경우 모두 파라미터 그룹의 이름, 패밀리 및 설명이 나열됩니다.

Example

다음 샘플 코드는 최대 20개의 파라미터 그룹을 나열합니다.

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeParameterGroups
   &MaxResults=20
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20210802T192317Z
   &Version=2021-01-01
   &X-Amz-Credential=<credential>
```

이 작업의 응답은 각 파라미터 그룹에 대해 memorydb_redis6의 경우 이름, 패밀리 및 설명을 나열하는 것과 같습니다.

```
<DescribeParameterGroupsResponse xmlns="http://memory-db.us-east-1.amazonaws.com/
doc/2021-01-01/">
  <DescribeParameterGroupsResult>
    <ParameterGroups>
      <ParameterGroup>
        <Name>myRedis6x</Name>
        <Family>memorydb_redis6</Family>
        <Description>My custom Redis OSS 6 parameter group</Description>
        <ARN>arn:aws:memorydb:us-east-1:012345678912:parametergroup/myredis6x</ARN>
      </ParameterGroup>
       <ParameterGroup>
        <Name>default.memorydb-redis6</Name>
        <Family>memorydb_redis6</Family>
        <Description>Default parameter group for memorydb_redis6</Description>
        <ARN>arn:aws:memorydb:us-east-1:012345678912:parametergroup/default.memorydb-
redis6</ARN>
      </ParameterGroup>
    </ParameterGroups>
  </DescribeParameterGroupsResult>
  <ResponseMetadata>
    <RequestId>3540cc3d-af48-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeParameterGroupsResponse>
```

Example

다음 샘플 코드에는 myRedis6x 파라미터 그룹이 나열되어 있습니다.

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeParameterGroups
   &ParameterGroupName=myRedis6x
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20210802T192317Z
   &Version=2021-01-01
   &X-Amz-Credential=<credential>
```

이 작업의 응답은 각 파라미터 그룹의 이름, 패밀리 및 설명을 나열하는 것과 같습니다.

```
<DescribeParameterGroupsResponse xmlns="http://memory-db.us-east-1.amazonaws.com/
doc/2021-01-01/">
  <DescribeParameterGroupsResult>
```

```
    <ParameterGroups>
      <ParameterGroup>
        <Name>myRedis6x</Name>
        <Family>memorydb_redis6</Family>
        <Description>My custom Redis OSS 6 parameter group</Description>
        <ARN>arn:aws:memorydb:us-east-1:012345678912:parametergroup/myredis6x</ARN>
      </ParameterGroup>
    </ParameterGroups>
  </DescribeParameterGroupsResult>
  <ResponseMetadata>
    <RequestId>3540cc3d-af48-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeParameterGroupsResponse>
```

자세한 내용은 DescribeParameterGroups 단원을 참조하십시오.

파라미터 그룹의 값 목록 조회

MemoryDB 콘솔AWS CLI, 또는 MemoryDB API를 사용하여 파라미터 및 파라미터 그룹의 값을 나열할 수 있습니다.

파라미터 그룹의 값 목록 조회(콘솔)

다음 절차에서는 MemoryDB 콘솔을 사용하여 파라미터 및 파라미터 그룹의 값을 나열하는 방법을 보여줍니다.

MemoryDB 콘솔을 사용하여 파라미터 그룹의 파라미터 및 그 값을 나열하려면

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/memorydb/ MemoryDB 콘솔을 엽니다.
2. 사용 가능한 모든 파라미터 목록을 표시하려면 왼쪽 탐색 창에서 [Parameter Groups]를 선택합니다.
3. 파라미터 그룹의 이름의 이름(옆에 있는 상자가 아님)을 선택하여 파라미터 및 값을 나열할 파라미터 그룹을 선택합니다.

   파라미터 및 그 값이 화면 하단에 나열됩니다. 파라미터의 수가 많으면 위아래로 스크롤하여 원하는 파라미터를 찾습니다.

파라미터 그룹의 값 나열(AWSCLI)

를 사용하여 파라미터 그룹의 파라미터와 해당 값을 나열하려면 명령을 AWS CLI사용합니다`describe-parameters`.

Example

다음 샘플 코드는 myRedis6x 파라미터 그룹의 모든 파라미터 및 그 값을 나열합니다.

Linux, macOS, Unix의 경우:

```
aws memorydb describe-parameters \
    --parameter-group-name myRedis6x
```

Windows의 경우:

```
aws memorydb describe-parameters ^
```

```
    --parameter-group-name myRedis6x
```

자세한 내용은 [describe-parameters](#) 단원을 참조하십시오.

파라미터 그룹 값 나열(MemoryDB API)

MemoryDB API를 사용하여 파라미터 그룹의 파라미터 및 그 값을 나열하려면 `DescribeParameters` 작업을 사용합니다.

자세한 내용은 [DescribeParameters](#) 단원을 참조하십시오.

파라미터 그룹 수정

> ⚠ Important
>
> 어떤 기본 파라미터 그룹도 수정할 수 없습니다.

파라미터 그룹의 일부 파라미터 값을 수정할 수 있습니다. 이 파라미터 값은 파라미터 그룹과 연결된 클러스터에 적용됩니다. 변경한 파라미터 값이 파라미터 그룹에 적용되는 시점에 관한 자세한 내용은 [엔진별 파라미터](#) 섹션을 참조하세요.

파라미터 그룹 수정(콘솔)

다음 절차는 MemoryDB 콘솔을 사용하여 파라미터 값을 변경하는 방법을 보여줍니다. 동일한 절차를 통해 모든 파라미터 값을 변경할 수 있습니다.

MemoryDB 콘솔을 사용하여 파라미터 값을 변경하려면

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/memorydb/ MemoryDB 콘솔을 엽니다.
2. 사용 가능한 모든 파라미터 목록을 표시하려면 왼쪽 탐색 창에서 [Parameter Groups]를 선택합니다.
3. 파라미터 그룹의 이름 왼쪽에 있는 라디오 버튼을 선택하여 수정할 파라미터 그룹을 선택합니다.

   작업을 선택한 다음 세부 정보 보기를 선택합니다. 또는 파라미터 그룹 이름을 선택하여 세부 정보 페이지로 이동할 수도 있습니다.
4. 파라미터를 수정하려면 편집을 선택합니다. 편집 가능한 모든 파라미터를 편집할 수 있게 됩니다. 변경하려는 파라미터를 찾으려면 페이지를 이동해야 할 수도 있습니다. 또는 검색 상자에서 이름, 값 또는 유형으로 파라미터를 검색할 수 있습니다.

5. 필요에 따라 파라미터를 수정합니다.
6. 변경 사항을 저장하려면 변경 사항 저장을 선택합니다.
7. 여러 페이지의 파라미터 값을 수정한 경우 변경 내용 미리 보기를 선택하여 모든 변경 내용을 검토할 수 있습니다. 변경을 확인하려면 변경 저장을 선택합니다. 추가로 수정하려면 뒤로를 선택합니다.
8. 파라미터 세부 정보 페이지에는 기본값으로 재설정하는 옵션도 있습니다. 기본값으로 재설정하려면 기본값으로 재설정을 선택합니다. 체크박스는 모든 파라미터의 왼쪽에 나타납니다. 재설정하려는 항목을 선택하고 재설정 진행을 선택하여 확인할 수 있습니다.

   대화 상자에서 확인을 선택하여 재설정 작업을 확인합니다.
9. 파라미터 세부 정보 페이지에서는 각 페이지에 표시할 파라미터 수를 설정할 수 있습니다. 오른쪽에 있는 톱니바퀴를 사용하여 변경하세요. 세부 정보 페이지에서 원하는 열을 활성화/비활성화할 수도 있습니다. 이러한 변경 사항은 콘솔 세션 내내 지속됩니다.

   변경한 파라미터의 이름을 찾으려면 엔진별 파라미터 섹션을 참조하세요.

파라미터 그룹 수정(AWSCLI)

를 사용하여 파라미터의 값을 변경하려면 명령을 AWS CLI사용합니다`update-parameter-group`.

변경하려는 파라미터의 이름과 허용되는 값을 찾으려면 엔진별 파라미터 섹션을 참조하세요.

자세한 정보는 update-parameter-group 단원을 참조하세요.

파라미터 그룹 수정(MemoryDB API)

MemoryDB API를 사용하여 파라미터 그룹의 파라미터 값을 변경하려면 `UpdateParameterGroup` 작업을 사용합니다.

변경하려는 파라미터의 이름과 허용되는 값을 찾으려면 엔진별 파라미터 섹션을 참조하세요.

자세한 내용은 `UpdateParameterGroup` 단원을 참조하십시오.

파라미터 그룹 삭제

MemoryDB 콘솔AWS CLI, 또는 MemoryDB API를 사용하여 사용자 지정 파라미터 그룹을 삭제할 수 있습니다.

파라미터 그룹이 클러스터와 연결된 경우 삭제할 수 없습니다. 또한 기본 파라미터 그룹도 삭제할 수 없습니다.

파라미터 그룹 삭제(콘솔)

다음 절차는 MemoryDB 콘솔을 사용하여 파라미터 그룹을 삭제하는 방법을 보여줍니다.

MemoryDB 콘솔을 사용하여 파라미터 그룹을 삭제하려면

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/memorydb/ MemoryDB 콘솔을 엽니다.
2. 사용 가능한 모든 파라미터 목록을 표시하려면 왼쪽 탐색 창에서 [Parameter Groups]를 선택합니다.
3. 파라미터 그룹의 이름 왼쪽에 있는 라디오 버튼을 선택하여 삭제할 파라미터 그룹을 선택합니다.

   Actions를 선택하고 삭제를 선택합니다.
4. Delete Parameter Groups 확인 화면이 나타납니다.
5. 파라미터 그룹을 삭제하려면 확인 텍스트 상자에 삭제를 입력합니다.

   파라미터 그룹을 유지하려면 취소를 선택합니다.

파라미터 그룹 삭제(AWSCLI)

를 사용하여 파라미터 그룹을 삭제하려면 명령을 AWS CLI사용합니다`delete-parameter-group`. 삭제할 파라미터 그룹의 경우 `--parameter-group-name`으로 지정된 파라미터 그룹에는 클러스터를 연결할 수 없으며 기본 파라미터 그룹이 될 수도 없습니다.

다음 샘플 코드에서는 myRedis6x 파라미터 그룹을 삭제합니다.

Example

Linux, macOS, Unix의 경우:

```
aws memorydb delete-parameter-group \
```

```
    --parameter-group-name myRedis6x
```

Windows의 경우:

```
aws memorydb delete-parameter-group ^
    --parameter-group-name myRedis6x
```

자세한 정보는 [delete-parameter-group](#) 단원을 참조하세요.

파라미터 그룹 삭제(MemoryDB API)

MemoryDB API를 사용하여 파라미터 그룹을 삭제하려면 `DeleteParameterGroup` 작업을 사용합니다. 삭제할 파라미터 그룹의 경우 `ParameterGroupName`으로 지정된 파라미터 그룹에는 클러스터를 연결할 수 없으며 기본 파라미터 그룹이 될 수도 없습니다.

Example

다음 샘플 코드에서는 myRedis6x 파라미터 그룹을 삭제합니다.

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DeleteParameterGroup
   &ParameterGroupName=myRedis6x
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20210802T192317Z
   &Version=2021-01-01
   &X-Amz-Credential=<credential>
```

자세한 내용은 [`DeleteParameterGroup`](#) 단원을 참조하십시오.

엔진별 파라미터

Valkey 또는 Redis OSS 클러스터에 파라미터 그룹을 지정하지 않으면 엔진 버전에 적절한 기본 파라미터 그룹이 사용됩니다. 기본 파라미터 그룹에서는 어떤 파라미터 값도 변경할 수 없습니다. 그러나 조건부로 수정 가능한 파라미터 값이 두 파라미터 그룹에서 동일하면 언제든지 사용자 지정 파라미터 그룹을 생성하고 클러스터에 할당할 수 있습니다. 자세한 내용은 파라미터 그룹 생성 단원을 참조하십시오.

주제

- Valkey 7 및 Redis OSS 7 파라미터 변경 사항
- Redis OSS 6 파라미터
- MemoryDB 노드 유형별 파라미터

Valkey 7 및 Redis OSS 7 파라미터 변경 사항

> **Note**
>
> MemoryDB는 변경 불가능한 새로운 파라미터 그룹 `default.memorydb-valkey7.search`를 포함하는 벡터 검색을 도입했습니다. 이 파라미터 그룹은 MemoryDB 콘솔에서 사용할 수 있으며 create-cluster CLI 명령을 사용하여 벡터 검색이 가능한 새 클러스터를 생성할 때 사용할 수 있습니다. 미리 보기 릴리스는 미국 동부(버지니아 북부), 미국 동부(오하이오), 미국 서부(오레곤), 아시아 태평양(도쿄), 유럽(아일랜드) AWS리전에서 사용할 수 있습니다.

파라미터 그룹 패밀리: memorydb_valkey7

Valkey 7 및 Redis OSS 7에 추가된 파라미터는 다음과 같습니다.

이름	세부 정보	설명
`latency-tracking`	허용되는 값: yes, no 기본값: no 유형: 문자열	yes로 설정하면 명령별 지연 시간을 추적하고 INFO 지연 시간 통계 명령을 통해 백분위수 분포 내보내기를 활성화하며, LATENCY 명령을 통해 지연 시간 분포(히스토그램)를 누적 집계합니다.

이름	세부 정보	설명
	수정 가능 여부: 예 변경 적용: 클러스터의 모든 노드에 즉시 적용됨	
`hash-max-listpack-entries`	허용되는 값: `0+` 기본값: 512 유형: 정수 수정 가능 여부: 예 변경 적용: 클러스터의 모든 노드에 즉시 적용됨	데이터세트를 압축하기 위한 해시 항목 최대 개수입니다.
`hash-max-listpack-value`	허용되는 값: `0+` 기본값: 64 유형: 정수 수정 가능 여부: 예 변경 적용: 클러스터의 모든 노드에 즉시 적용됨	데이터세트를 압축하기 위한 최대 해시 항목의 임곗값입니다.
`zset-max-listpack-entries`	허용되는 값: `0+` 기본값: 128 유형: 정수 수정 가능 여부: 예 변경 적용: 클러스터의 모든 노드에 즉시 적용됨	데이터세트를 압축하기 위한 정렬된 세트 항목 최대 개수입니다.

이름	세부 정보	설명
`zset-max-listpack-value`	허용되는 값: `0+` 기본값: 64 유형: 정수 수정 가능 여부: 예 변경 적용: 클러스터의 모든 노드에 즉시 적용됨	데이터세트를 압축하기 위한 정렬된 최대 세트 항목의 임곗값입니다.
`search-enabled`	허용되는 값: `yes, no` 기본값: no 유형: 문자열 수정 가능 여부: 예 변경 사항 적용: 새 클러스터에만 해당됩니다. 최소 엔진 버전: 7.1	yes로 설정하면 검색 기능이 활성화됩니다.
`search-query-timeout-ms`	허용되는 값: `1 - 60,000` 기본값: `10,000` 유형: 정수 수정 가능 여부: 예 변경 적용: 클러스터의 모든 노드에 즉시 적용됨 최소 엔진 버전: 7.1	검색 쿼리가 실행될 수 있는 밀리초 단위의 최대 시간입니다.

Redis OSS 7에서 변경된 파라미터는 다음과 같습니다.

이름	세부 정보	설명
activerehashing	수정 가능: no. Redis OSS 7에서는 이 파라미터가 기본적으로 숨겨져 있고 활성화되어 있습니다. 비활성화하려면 지원 사례를 생성해야 합니다.	수정 가능 여부는 '예'였습니다.

Redis OSS 7에서 제거된 파라미터는 다음과 같습니다.

이름	세부 정보	설명
hash-max-ziplist-entries	허용되는 값: 0+ 기본값: 512 유형: 정수 수정 가능 여부: 예 변경 적용: 클러스터의 모든 노드에 즉시 적용됨	작은 해시 인코딩을 표현하는 데 ziplist 대신 listpack 사용
hash-max-ziplist-value	허용되는 값: 0+ 기본값: 64 유형: 정수 수정 가능 여부: 예 변경 적용: 클러스터의 모든 노드에 즉시 적용됨	작은 해시 인코딩을 표현하는 데 ziplist 대신 listpack 사용

이름	세부 정보	설명
zset-max-ziplist-entries	허용되는 값: 0+ 기본값: 128 유형: 정수 수정 가능 여부: 예 변경 적용: 클러스터의 모든 노드에 즉시 적용됨	작은 해시 인코딩을 표현하는 데 ziplist 대신 listpack 사용.
zset-max-ziplist-value	허용되는 값: 0+ 기본값: 64 유형: 정수 수정 가능 여부: 예 변경 적용: 클러스터의 모든 노드에 즉시 적용됨	작은 해시 인코딩을 표현하는 데 ziplist 대신 listpack 사용.

Redis OSS 6 파라미터

> **Note**
>
> Redis OSS 버전 6.2에서는 데이터 계층화에 사용하기 위해 r6gd 노드 패밀리가 도입되었을 때 noeviction, volatile-lru 및 allkeys-lru max-memory 정책만 r6gd 노드 유형에서 지원됩니다.

파라미터 그룹 패밀리: memorydb_redis6

Redis OSS 6에 추가된 파라미터는 다음과 같습니다.

이름	세부 정보	설명
`maxmemory-policy`	유형: 문자열 허용된 값: volatile-lru,allkeys-lru,volatile-lfu,allkeys-lfu,volatile-random,allkeys-random,volatile-ttl,noeviction 기본값: noeviction	최대 메모리 사용량에 도달했을 때 키에 대한 제거 정책입니다. LRU 캐시로 Valkey 또는 Redis OSS 사용에 대한 자세한 내용은 키 제거를 참조하세요.
`list-compress-depth`	유형: 정수 허용되는 값: 0- 기본값: 0	압축 깊이는 압축에서 제외할 목록 각 측면의 퀵리스트 집리스트 노드 수입니다. 목록의 헤드와 테일은 빠른 푸시 및 팝 작업을 위해 항상 압축하지 않습니다. 설정: • 0: 모든 압축을 해제합니다. • 1: 헤드와 테일에서 첫 번째 노드로 압축을 시작합니다. [헤드] -> 노드 -> 노드 -> ... -> 노드 -> [테일] [헤드]와 [테일]을 제외한 모든 노드를 압축합니다. • 2: 헤드와 테일에서 두 번째 노드로 압축을 시작합니다. [헤드] -> [다음] -> 노드 -> 노드>... -> 노드 -> [이전] -> [테일] [헤드], [다음], [이전], [테일]은 압축하지 않습니다. 다른 모든 노드를 압축합니다. • 기타.
`hll-sparse-max-bytes`	유형: 정수 허용되는 값: 1~16000	HyperLogLog 스파스 표현 바이트 제한입니다. 제한은 16바이트 헤더를 포함합니다. 스파스 표

이름	세부 정보	설명
	기본값: 3000	현을 사용하는 HyperLogLog가 이 제한을 초과하면 밀도가 높은 표현으로 변환됩니다. 그 시점에서는 밀도가 높은 표현이 메모리 효율을 높이기 때문에 16000보다 큰 값은 권장하지 않습니다. 스파스 인코딩이 너무 많은 O(N)인 `PFADD`를 너무 느리게 하지 않고 공간 효율적인 인코딩의 이점을 얻으려면 값을 약 3000까지로 하는 것이 좋습니다. CPU의 문제가 아니라 공백이 있고 데이터세트가 0~15000 범위의 카디널리티(cardinality)를 가진 많은 HyperLogLog로 구성되어 있으면 값을 10000까지 올릴 수 있습니다.
`lfu-log-factor`	유형: 정수 허용되는 값: 1- 기본값: 10	LFU 제거 정책의 키 카운터 증가를 위한 로그 팩터.
`lfu-decay-time`	유형: 정수 허용되는 값: 0- 기본값: 1	LFU 제거 정책에 대한 키 카운터 감소에 소요된 시간입니다(분).
`active-defrag-max-scan-fields`	유형: 정수 허용되는 값: 1~1000000 기본값: 1000	활성 조각 모음 중 기본 사전 스캔에서 처리될 최대 set/hash/zset/list 필드 수입니다.

이름	세부 정보	설명
active-defrag-threshold-upper	유형: 정수 허용되는 값: 1~100 기본값: 100	최대 작업을 사용하는 조각의 최대 비율입니다.
client-output-buffer-limit-pubsub-hard-limit	유형: 정수 허용되는 값: 0- 기본값: 33554432	Redis OSS 게시/구독 클라이언트: 클라이언트의 출력 버퍼가 특정 바이트 수에 도달하면 클라이언트가 연결 해제됩니다.
client-output-buffer-limit-pubsub-soft-limit	유형: 정수 허용되는 값: 0- 기본값: 8388608	Redis OSS 게시/구독 클라이언트: 클라이언트의 출력 버퍼가 특정 바이트 수에 도달하면 클라이언트가 연결 해제됩니다. 그러나 이러한 조건은 `client-output-buffer-limit-pubsub-soft-seconds.` 의 경우에만 지속됩니다.
client-output-buffer-limit-pubsub-soft-seconds	유형: 정수 허용되는 값: 0- 기본값: 60	Redis OSS 게시/구독 클라이언트: 클라이언트의 출력 버퍼가 `client-output-buffer-limit-pubsub-soft-limit` 바이트에 해당 시간(초)보다 오래 유지되면 클라이언트가 연결 해제됩니다.
timeout	유형: 정수 허용되는 값: 0,20- 기본값: 0	제한 시간이 지나기 전에 노드가 대기하는 시간(초)입니다. 유효한 값: • 0 – 유휴 클라이언트 연결을 절대로 끊지 마세요. • 1~19 - 잘못된 값입니다. • >=20 – 유휴 클라이언트 연결을 끊기 전에 노드가 대기하는 시간(초)입니다.

이름	세부 정보	설명
`notify-keyspace-events`	유형: 문자열 허용되는 값: N/A 기본값: NULL	Redis OSS가 Pub/Sub 클라이언트에 알리기 위한 키스페이스 이벤트입니다. 기본적으로 모든 알림은 비활성화되어 있습니다.
`maxmemory-samples`	유형: 정수 허용되는 값: 1- 기본값: 3	가장 오랫동안 사용되지 않은 (`LRU`) 및 `time-to-live` (`TTL`) 계산의 경우 파라미터는 확인할 키의 샘플 크기를 나타냅니다. 기본적으로 Redis OSS는 키 3개를 선택하고 가장 최근에 사용한 키를 사용합니다.
`slowlog-max-len`	유형: 정수 허용되는 값: 0- 기본값: 128	Redis OSS 슬로우 로그의 최대 길이입니다. 이 길이에는 제한이 없습니다. 단, 메모리를 소모한다는 점만 알아두세요. `SLOWLOG RESET.`을 사용하면 슬로우 로그에서 사용한 메모리를 회수할 수 있습니다.
`activerehashing`	유형: 문자열 허용되는 값: yes/no 기본값: yes	기본 해시 테이블은 초당 10회 재해시되며 각 해시 작업은 1밀리초의 CPU 시간을 소비합니다. 이 값은 파라미터 그룹을 생성할 때 설정됩니다. 새 파라미터 그룹을 클러스터에 할당할 때 이전 파라미터 그룹과 새 파라미터 그룹에서 값이 동일해야 합니다.
`client-output-buffer-limit-normal-hard-limit`	유형: 정수 허용되는 값: 0- 기본값: 0	클라이언트의 출력 버퍼가 특정 바이트 수에 도달하면 클라이언트가 연결 해제됩니다. 기본값은 0입니다(하드 제한 없음).

이름	세부 정보	설명
client-output-buffer-limit-normal-soft-limit	유형: 정수 허용되는 값: 0- 기본값: 0	클라이언트의 출력 버퍼가 특정 바이트 수에 도달하면 클라이언트가 연결 해제됩니다. 그러나 이러한 조건은 `client-output-buffer-limit-normal-soft-seconds` 의 경우에만 지속됩니다. 기본값은 0입니다(소프트 제한 없음).
client-output-buffer-limit-normal-soft-seconds	유형: 정수 허용되는 값: 0- 기본값: 0	클라이언트의 출력 버퍼가 `client-output-buffer-limit-normal-soft-limit` 바이트에 해당 시간(초)보다 오래 유지되면 클라이언트가 연결 해제됩니다. 기본값은 0입니다(시간 제한 없음).
tcp-keepalive	유형: 정수 허용되는 값: 0- 기본값: 300	이 값을 0이 아닌 값(N)으로 설정하면 연결이 되어있는지 확인하기 위해 노드 클라이언트가 N초마다 폴링됩니다. 기본 설정인 0을 사용하면 폴링이 발생하지 않습니다.
active-defrag-cycle-min	유형: 정수 허용되는 값: 1~75 기본값: 5	CPU 비율의 조각 모음에 대한 최소 작업입니다.
stream-node-max-bytes	유형: 정수 허용되는 값: 0- 기본값: 4096	스트림 데이터 구조는 내부에 여러 항목을 인코딩하는 노드의 기수 트리입니다. 이 구성을 사용하여 기수 트리에서 단일 노드의 최대 크기를 바이트로 지정합니다. 0으로 설정하면 트리 노드의 크기는 무제한입니다.

이름	세부 정보	설명
stream-node-max-entries	유형: 정수 허용되는 값: 0- 기본값: 100	스트림 데이터 구조는 내부에 여러 항목을 인코딩하는 노드의 기수 트리입니다. 이 구성을 사용하여 새 노드 항목을 추가할 때 새 노드로 전환하기 전에 단일 노드에 포함할 수 있는 최대 항목 수를 지정합니다. 0으로 설정하면 트리 노드의 항목 수는 무제한입니다.
lazyfree-lazy-eviction	유형: 문자열 허용되는 값: yes/no 기본값: 아니요	제거 시 비동기식 삭제를 수행합니다.
active-defrag-ignore-bytes	유형: 정수 허용되는 값: 1048576- 기본값: 104857600	활성 조각 모음을 시작하는 조각의 최소 수입니다.
lazyfree-lazy-expire	유형: 문자열 허용되는 값: yes/no 기본값: 아니요	키 만료 시 비동기식 삭제를 수행합니다.
active-defrag-threshold-lower	유형: 정수 허용되는 값: 1~100 기본값: 10	활성 조각 모음을 시작하는 조각의 최소 비율입니다.

이름	세부 정보	설명
`active-defrag-cycle-max`	유형: 정수 허용되는 값: 1~75 기본값: 75	CPU 비율의 조각 모음에 대한 최대 작업입니다.
`lazyfree-lazy-server-del`	유형: 문자열 허용되는 값: yes/no 기본값: 아니요	값을 업데이트하는 명령에 대해 비동기식 삭제를 수행합니다.
`slowlog-log-slower-than`	유형: 정수 허용되는 값: 0- 기본값: 10000	Redis OSS `Slow Log` 기능에서 명령을 로깅하기 위해 초과되어야 하는 최대 실행 시간(마이크로초)입니다. 참고로 음수는 느린 로그를 비활성화하고, 값이 0이면 모든 명령의 로깅을 강제합니다.
`hash-max-ziplist-entries`	유형: 정수 허용되는 값: 0- 기본값: 512	해시에 사용되는 메모리 양을 결정합니다. 지정된 수보다 적은 수의 항목이 있는 해시는 공간을 절약하는 특수 인코딩을 사용하여 저장됩니다.
`hash-max-ziplist-value`	유형: 정수 허용되는 값: 0- 기본값: 64	해시에 사용되는 메모리 양을 결정합니다. 지정된 바이트 수보다 작은 항목이 있는 해시는 공간을 절약하는 특수 인코딩을 사용하여 저장됩니다.

이름	세부 정보	설명
set-max-intset-entries	유형: 정수 허용되는 값: 0- 기본값: 512	특정 종류의 세트(64비트 부호가 있는 정수의 범위에서 기수 10의 정수 문자열)에 사용되는 메모리의 양을 결정합니다. 지정된 수보다 적은 수의 항목이 있는 세트는 공간을 절약하는 특수 인코딩을 사용하여 저장됩니다.
zset-max-ziplist-entries	유형: 정수 허용되는 값: 0- 기본값: 128	정렬된 세트에 사용할 메모리 양을 결정합니다. 지정된 수보다 적은 수의 정렬된 세트는 공간을 절약하는 특수 인코딩을 사용하여 저장됩니다.
zset-max-ziplist-value	유형: 정수 허용되는 값: 0- 기본값: 64	정렬된 세트에 사용할 메모리 양을 결정합니다. 지정된 바이트 수보다 작은 항목이 있는 정렬된 세트는 공간을 절약하는 특수 인코딩을 사용하여 저장됩니다.
tracking-table-max-keys	유형: 정수 허용되는 값: 1~100000000 기본값: 1000000	클라이언트 측 캐싱을 지원하기 위해 Redis OSS는 어떤 클라이언트가 어떤 키에 액세스했는지 확인하는 추적을 지원합니다. 추적된 키가 수정되면 무효화 메시지가 모든 클라이언트에 전송되어 키의 캐시된 값이 더 이상 유효하지 않음을 알립니다. 이 값을 사용하면 이 테이블의 상한을 지정할 수 있습니다.
acllog-max-len	유형: 정수 허용되는 값: 1~10000 기본값: 128	ACL 로그의 최대 항목 수입니다.

이름	세부 정보	설명
`active-expire-effort`	유형: 정수 허용되는 값: 1~10 기본값: 1	Redis OSS는 두 가지 메커니즘을 사용하여 유지 시간(TTL)이 초과된 키를 삭제합니다. 하나는 키가 액세스되고 만료된 것으로 확인된 경우입니다. 다른 하나는 정기적인 작업이 키를 샘플링하고 유지 시간(TTL)이 초과된 키를 만료시키는 경우입니다. 이 파라미터는 Redis OSS가 정기 작업에서 항목을 만료시키는 데 사용하는 작업량을 정의합니다. 기본값 1은 만료된 키의 10% 이상이 메모리에 남아 있지 않도록 합니다. 또한 총 메모리의 25% 이상을 소비하지 않도록 시스템에 대기 시간을 추가합니다. 이 값을 최대 10까지 늘려 키 만료에 사용되는 작업량을 늘릴 수 있습니다. CPU 사용량이 늘어나고 대기 시간이 길어진다는 단점이 있습니다. 높은 메모리 사용량과 CPU 사용률 증가를 허용할 수 있는 경우가 아니면 1의 값을 사용하는 것이 좋습니다.
`lazyfree-lazy-user-del`	유형: 문자열 허용되는 값: yes/no 기본값: 아니요	`DEL` 명령의 기본 동작이 `UNLINK`와 동일하게 작동하는지 여부를 지정합니다.
`activedefrag`	유형: 문자열 허용되는 값: yes/no 기본값: 아니요	활성 메모리 조각 모음 파라미터가 활성화됩니다.

이름	세부 정보	설명
`maxclients`	유형: 정수 허용되는 값: 65000 기본값: 65000	한 번에 연결할 수 있는 최대 클라이언트 수입니다. 이는 수정할 수 없습니다.
`client-query-buffer-limit`	유형: 정수 허용되는 값: 1,048,576 ~1,073,741,824 기본값: 1073741824	단일 클라이언트 쿼리 버퍼의 최대 크기입니다. 변경 사항은 즉시 적용됩니다.
`proto-max-bulk-len`	유형: 정수 허용되는 값: 1,048,576 ~536,870,912 기본값: 536870912	단일 요소 요청의 최대 크기입니다. 변경 사항은 즉시 적용됩니다.

MemoryDB 노드 유형별 파라미터

대부분의 파라미터는 단일 값을 갖지만 일부 파라미터는 사용하는 노드 유형에 따라 다양한 값을 갖습니다. 다음 표에는 각 노드 유형에 대한 `maxmemory`의 기본값이 나와 있습니다. `maxmemory`의 값은 노드에서 데이터 및 다른 용도에 사용할 수 있는 최대 바이트 수입니다.

노드 유형	Maxmemory
db.r7g.large	14037181030
db.r7g.xlarge	28261849702
db.r7g.2xlarge	56711183565
db.r7g.4xlarge	113609865216

노드 유형	Maxmemory
db.r7g.8xlarge	225000375228
db.r7g.12xlarge	341206346547
db.r7g.16xlarge	450000750456
db.r6gd.xlarge	28261849702
db.r6gd.2xlarge	56711183565
db.r6gd.4xlarge	113609865216
db.r6gd.8xlarge	225000375228
db.r6g.large	14037181030
db.r6g.xlarge	28261849702
db.r6g.2xlarge	56711183565
db.r6g.4xlarge	113609865216
db.r6g.8xlarge	225000375228
db.r6g.12xlarge	341206346547
db.r6g.16xlarge	450000750456
db.t4g.small	1471026299
db.t4g.medium	3317862236

Note

모든 MemoryDB 인스턴스 유형은 Amazon Virtual Private Cloud VPC에서 생성해야 합니다.

제한된 명령

관리형 서비스 환경을 제공하기 위해 MemoryDB는 고급 권한이 필요한 특정 명령에 대한 액세스를 제한합니다. 다음과 같은 명령을 사용할 수 없습니다.

- `acl deluser`
- `acl load`
- `acl save`
- `acl setuser`
- `bgrewriteaof`
- `bgsave`
- `cluster addslot`
- `cluster delslot`
- `cluster setslot`
- `config`
- `debug`
- `migrate`
- `module`
- `psync`
- `replicaof`
- `save`
- `shutdown`
- `slaveof`
- `sync`

자습서: Amazon VPC에서 MemoryDB에 액세스하도록 Lambda 함수 구성

이 자습서에서는 다음을 수행하는 방법을 알아봅니다.

- us-east-1 리전 내 기본 Amazon Virtual Private Cloud(Amazon VPC)에서 MemoryDB 클러스터를 생성합니다.

- 이 클러스터에 액세스하기 위해 Lambda 함수를 생성합니다. Lambda 함수를 생성할 때 Amazon VPC 및 VPC 보안 그룹에 서브넷 ID를 제공하여 Lambda 함수가 VPC의 리소스에 액세스할 수 있도록 합니다. 이 자습서의 설명에서는 Lambda 함수가 UUID를 생성하여 이를 클러스터에 기록한 다음, 클러스터에서 이를 검색합니다.
- Lambda 함수를 수동으로 간접적으로 호출하고 VPC의 클러스터에 액세스했는지 확인합니다.
- 이 자습서를 위해 설정된 Lambda 함수, 클러스터 및 IAM 역할을 정리합니다.

주제

1단계: 클러스터 생성

클러스터를 생성하려면 아래 단계를 따르세요.

클러스터 생성

이 단계에서는 AWS Command Line Interface(CLI)를 사용하여 계정의 us-east-1 리전에 있는 기본 Amazon VPC에 클러스터를 생성합니다. MemoryDB 콘솔 또는 API를 사용한 클러스터 생성에 대한 자세한 내용은 2단계: 클러스터 생성 섹션을 참조하세요.

```
aws memorydb create-cluster --cluster-name cluster-01 --engine-version 7.0 --acl-name
 open-access \
--description "MemoryDB IAM auth application" \
--node-type db.r6g.large
```

상태 필드 값은 `CREATING`으로 설정됩니다. MemoryDB가 클러스터 생성을 완료하기까지 몇 분 정도 걸릴 수 있습니다.

클러스터 엔드포인트 복사

MemoryDB가 `describe-clusters` 명령을 사용하여 클러스터 생성을 완료했는지 확인합니다.

```
aws memorydb describe-clusters \
--cluster-name cluster-01
```

출력에 표시된 클러스터 엔드포인트 주소를 복사합니다. Lambda 함수의 배포 패키지를 생성할 때 이 주소가 필요합니다.

IAM 역할 생성

1. 아래에서와 같이, 역할에 IAM 신뢰 정책 문서를 생성하여 계정이 새 역할을 수임할 수 있도록 합니다. 정책을 trust-policy.json 파일에 저장합니다. 이 정책의 account_id 123456789012를 해당 account_id로 바꿔야 합니다.

   JSON

```
{
"Version":"2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": { "AWS": "arn:aws:iam::123456789012:root" },
        "Action": "sts:AssumeRole"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }]
}
```

2. 아래에서와 같이, IAM 정책 문서를 생성합니다. 정책을 policy.json 파일에 저장합니다. 이 정책의 account_id 123456789012를 해당 account_id로 바꿔야 합니다.

   JSON

```
{
"Version":"2012-10-17",
  "Statement": [
    {
      "Effect" : "Allow",
      "Action" : [
        "memorydb:Connect"
      ],
```

```
        "Resource" : [
          "arn:aws:memorydb:us-east-1:123456789012:cluster/cluster-01",
          "arn:aws:memorydb:us-east-1:123456789012:user/iam-user-01"
        ]
      }
    ]
}
```

3. IAM 역할을 생성합니다.

```
aws iam create-role \
--role-name "memorydb-iam-auth-app" \
--assume-role-policy-document file://trust-policy.json
```

4. IAM 정책을 생성합니다.

```
aws iam create-policy \
  --policy-name "memorydb-allow-all" \
  --policy-document file://policy.json
```

5. IAM 정책을 역할에 연결합니다. 이 policy-arn의 account_id 123456789012를 해당 account_id로 바꿔야 합니다.

```
aws iam attach-role-policy \
 --role-name "memorydb-iam-auth-app" \
 --policy-arn "arn:aws:iam::123456789012:policy/memorydb-allow-all"
```

액세스 제어 목록(ACL) 생성

1. IAM가 활성화된 사용자를 새로 생성합니다.

```
aws memorydb create-user \
  --user-name iam-user-01 \
--authentication-mode Type=iam \
--access-string "on ~* +@all"
```

2. ACL을 생성하고 클러스터에 연결합니다.

```
aws memorydb create-acl \
  --acl-name iam-acl-01 \
  --user-names iam-user-01
```

```
aws memorydb update-cluster \
  --cluster-name cluster-01 \
  --acl-name iam-acl-01
```

2단계: Lambda 함수 생성

Lambda 함수를 생성하려면 다음 단계를 수행합니다.

배포 패키지 생성

이 자습서에서는 Lambda 함수에 대한 Python의 예제 코드를 제공합니다.

Python

다음의 예제 Python 코드는 항목을 읽어서 MemoryDB 클러스터에 기록합니다. 코드를 복사한 후 `app.py` 파일에 저장합니다. 코드의 `cluster_endpoint` 값을 이전 단계에서 복사한 엔드포인트 주소로 바꿔야 합니다.

```
from typing import Tuple, Union
from urllib.parse import ParseResult, urlencode, urlunparse

import botocore.session
import redis
from botocore.model import ServiceId
from botocore.signers import RequestSigner
from cachetools import TTLCache, cached
import uuid

class MemoryDBIAMProvider(redis.CredentialProvider):
    def __init__(self, user, cluster_name, region="us-east-1"):
        self.user = user
        self.cluster_name = cluster_name
        self.region = region

        session = botocore.session.get_session()
        self.request_signer = RequestSigner(
            ServiceId("memorydb"),
            self.region,
            "memorydb",
            "v4",
            session.get_credentials(),
```

```
            session.get_component("event_emitter"),
        )

    # Generated IAM tokens are valid for 15 minutes
    @cached(cache=TTLCache(maxsize=128, ttl=900))
    def get_credentials(self) -> Union[Tuple[str], Tuple[str, str]]:
        query_params = {"Action": "connect", "User": self.user}

        url = urlunparse(
            ParseResult(
                scheme="https",
                netloc=self.cluster_name,
                path="/",
                query=urlencode(query_params),
                params="",
                fragment="",
            )
        )
        signed_url = self.request_signer.generate_presigned_url(
            {"method": "GET", "url": url, "body": {}, "headers": {}, "context": {}},
            operation_name="connect",
            expires_in=900,
            region_name=self.region,
        )
        # RequestSigner only seems to work if the URL has a protocol, but
        # MemoryDB only accepts the URL without a protocol
        # So strip it off the signed URL before returning
        return (self.user, signed_url.removeprefix("https://"))

def lambda_handler(event, context):
    username = "iam-user-01" # replace with your user id
    cluster_name = "cluster-01" # replace with your cache name
    cluster_endpoint = "clustercfg.cluster-01.xxxxxx.memorydb.us-east-1.amazonaws.com"
 # replace with your cluster endpoint
    creds_provider = MemoryDBIAMProvider(user=username, cluster_name=cluster_name)
    redis_client = redis.Redis(host=cluster_endpoint, port=6379,
 credential_provider=creds_provider, ssl=True, ssl_cert_reqs="none")

    key='uuid'
    # create a random UUID - this will be the sample element we add to the cluster
    uuid_in = uuid.uuid4().hex
    redis_client.set(key, uuid_in)
    result = redis_client.get(key)
    decoded_result = result.decode("utf-8")
```

```
    # check the retrieved item matches the item added to the cluster and print
    # the results
    if decoded_result == uuid_in:
        print(f"Success: Inserted {uuid_in}. Fetched {decoded_result} from MemoryDB.")
    else:
        raise Exception(f"Bad value retrieved. Expected {uuid_in}, got
 {decoded_result}")

    return "Fetched value from MemoryDB"
```

이 코드는 Python `redis-py` 라이브러리를 사용하여 항목을 클러스터에 넣고 검색합니다. 이 코드는 `cachetools`를 사용하여 생성된 IAM 인증 토큰을 15분 동안 캐시합니다. `redis-py` 및 `cachetools`가 포함된 배포 패키지를 만들려면 다음 단계를 수행합니다.

`app.py` 소스 코드 파일이 포함된 프로젝트 디렉터리에서 폴더 패키지를 만들어 `redis-py` 및 `cachetools` 라이브러리를 설치합니다.

```
mkdir package
```

pip를 사용하여 `redis-py` 및 `cachetools`를 설치합니다.

```
pip install --target ./package redis
pip install --target ./package cachetools
```

`redis-py` 및 `cachetools` 라이브러리가 포함된 .zip 파일을 생성합니다. Linux 및 MacOS에서 다음 명령을 실행합니다. Windows에서 선호하는 zip 유틸리티를 사용하여 루트에 `redis-py` 및 `cachetools` 라이브러리가 포함된 .zip 파일을 생성합니다.

```
cd package
zip -r ../my_deployment_package.zip .
```

함수 코드를 .zip 파일에 추가합니다. Linux 및 macOS에서 다음 명령을 실행합니다. Windows에서 선호하는 zip 유틸리티를 사용하여 .zip 파일의 루트에 app.py를 추가합니다.

```
cd ..
zip my_deployment_package.zip app.py
```

IAM 역할(실행 역할) 생성

라는 AWS관리형 정책을 `AWSLambdaVPCAccessExecutionRole` 역할에 연결합니다.

```
aws iam attach-role-policy \
 --role-name "memorydb-iam-auth-app" \
 --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaVPCAccessExecutionRole"
```

배포 패키지 업로드(Lambda 함수 생성)

이 단계에서는 create-function AWS CLI명령을 사용하여 Lambda 함수(AccessMemoryDB)를 생성합니다.

배포 패키지 .zip 파일이 들어 있는 프로젝트 디렉터리에서 다음 Lambda CLI `create-function` 명령을 실행합니다.

역할 옵션의 경우 이전 단계에서 생성한 실행 역할의 ARN을 사용합니다. vpc-config에 기본 VPC의 서브넷과 기본 VPC의 보안 그룹 ID를 쉼표로 구분한 목록을 입력합니다. 이러한 값은 Amazon VPC 콘솔에서 확인할 수 있습니다. 기본 VPC의 서브넷을 찾으려면 VPCs 선택한 다음 AWS계정의 기본 VPC를 선택합니다. 이 VPC의 보안 그룹을 확인하려면 보안으로 이동하여 보안 그룹을 선택합니다. us-east-1 리전이 선택되어 있어야 합니다.

```
aws lambda create-function \
--function-name AccessMemoryDB  \
--region us-east-1 \
--zip-file fileb://my_deployment_package.zip \
--role arn:aws:iam::123456789012:role/memorydb-iam-auth-app \
--handler app.lambda_handler \
--runtime python3.12  \
--timeout 30 \
--vpc-config SubnetIds=comma-separated-vpc-subnet-ids,SecurityGroupIds=default-
security-group-id
```

3단계: Lambda 함수 테스트

이 단계에서는 invoke 명령을 사용하여 Lambda 함수를 수동으로 간접적으로 호출합니다. Lambda 함수는 실행될 때 UUID를 생성하고 이를 Lambda 코드에 지정된 ElastiCache 캐시에 기록합니다. 그런 다음 Lambda 함수는 캐시에서 해당 항목을 검색합니다.

1. 호출 명령을 사용하여 Lambda 함수(AccessMemoryDB)AWS Lambda를 호출합니다.

   ```
   aws lambda invoke \
   --function-name AccessMemoryDB  \
   --region us-east-1 \
   ```

```
output.txt
```

2. 다음과 같이 Lambda 함수가 성공적으로 실행되었는지 확인합니다.

   - output.txt 파일을 검토합니다.
   - CloudWatch 콘솔을 열고 함수(/aws/lambda/AccessRedis)의 로그 그룹을 선택하여 CloudWatch Logs에서 결과를 확인합니다. 로그 스트림의 출력은 다음과 유사해야 합니다.

     ```
     Success: Inserted 826e70c5f4d2478c8c18027125a3e01e. Fetched
      826e70c5f4d2478c8c18027125a3e01e from MemoryDB.
     ```

   - AWS Lambda콘솔에서 결과를 검토합니다.

4단계: 정리(선택 사항)

정리하려면 다음 단계를 수행합니다.

Lambda 함수 삭제

```
aws lambda delete-function \
 --function-name AccessMemoryDB
```

MemoryDB 클러스터 삭제

클러스터를 삭제합니다.

```
aws memorydb delete-cluster \
 --cluster-name cluster-01
```

사용자 및 ACL을 제거합니다.

```
aws memorydb delete-user \
 --user-id iam-user-01

aws memorydb delete-acl \
 --acl-name iam-acl-01
```

IAM 역할 및 정책 제거

```
aws iam detach-role-policy \
```

```
 --role-name "memorydb-iam-auth-app" \
 --policy-arn "arn:aws:iam::123456789012:policy/memorydb-allow-all"

aws iam detach-role-policy \
--role-name "memorydb-iam-auth-app" \
--policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaVPCAccessExecutionRole"

aws iam delete-role \
 --role-name "memorydb-iam-auth-app"

 aws iam delete-policy \
  --policy-arn "arn:aws:iam::123456789012:policy/memorydb-allow-all"
```

벡터 검색

MemoryDB에 대한 벡터 검색은 MemoryDB의 기능을 확장합니다. 벡터 검색은 기존 MemoryDB 기능과 함께 사용할 수 있습니다. 벡터 검색을 사용하지 않는 애플리케이션은 벡터 검색의 존재 여부에 영향을 받지 않습니다. MemoryDB를 사용할 수 있는 모든 리전에서 벡터 검색을 사용할 수 있습니다.

벡터 검색은 고속 벡터 검색을 제공하는 동시에 애플리케이션 아키텍처를 간소화합니다. MemoryDB의 벡터 검색은 최대 성능과 규모가 가장 중요한 선택 기준인 사용 사례에 적합합니다. 기존 MemoryDB 데이터나 Valkey 또는 Redis OSS API를 사용하여 기계 학습 및 생성형 AI 사용 사례를 구축할 수 있습니다. 여기에는 검색 증강 생성, 이상 탐지, 문서 검색, 실시간 권장 사항이 포함됩니다.

2024년 6월 26일 기준, AWS MemoryDB는 AWS의 널리 사용되는 벡터 데이터베이스 중 가장 높은 재현율로 가장 빠른 벡터 검색 성능을 제공합니다.

주제

벡터 검색 개요

벡터 검색은 인덱스 생성, 유지 관리 및 사용을 기반으로 합니다. 각 벡터 검색 작업은 단일 인덱스를 지정하며 해당 인덱스로만 제한됩니다. 즉, 한 인덱스에 대한 작업은 다른 인덱스에 대한 작업의 영향을 받지 않습니다. 인덱스 생성 및 삭제 작업을 제외하고 언제든지 모든 인덱스에 대해 원하는 수의 작업을 실행할 수 있습니다. 즉, 클러스터 수준에서는 여러 인덱스에 대한 여러 작업이 동시에 진행될 수 있습니다.

개별 인덱스는 키, 함수 등 다른 Valkey 및 Redis OSS 네임스페이스와는 별개인 고유한 네임스페이스에 존재하는 명명된 객체입니다. 각 인덱스는 열과 행이라는 두 가지 차원으로 구성된다는 점에서 기존 데이터베이스 테이블과 개념적으로 유사합니다. 테이블의 각 행은 키에 해당합니다. 인덱스의 각 열은 해당 키의 멤버 또는 일부에 해당합니다. 이 문서 내에서 키, 행 및 레코드라는 용어는 동일한 의미로 통용됩니다. 마찬가지로 열, 필드, 경로 및 멤버라는 용어는 본질적으로 동일한 의미로 통용됩니다.

인덱싱된 데이터를 추가, 삭제 또는 수정하기 위한 특별한 명령은 없습니다. 대신 인덱스에 있는 키를 수정하는 기존 HASH 또는 JSON 명령도 인덱스를 자동으로 업데이트합니다.

주제

인덱스와 Valkey 및 Redis OSS 키스페이스

인덱스는 Valkey 및 Redis OSS 키스페이스의 하위 집합을 기준으로 구성되고 유지 관리됩니다. 여러 인덱스는 키스페이스의 분리되거나 중복되는 하위 집합을 제한 없이 선택할 수 있습니다. 각 인덱스의 키스페이스는 인덱스 생성 시 제공되는 키 접두사 목록에 의해 정의됩니다. 접두사 목록은 선택 사항이며 생략하면 전체 키스페이스가 해당 인덱스에 포함됩니다. 또한 인덱스는 유형이 일치하는 키만 포함하도록 입력됩니다. 현재는 JSON 및 HASH 인덱스만 지원됩니다. HASH 인덱스는 접두사 목록에 포함된 HASH 키만 인덱싱하고, 마찬가지로 JSON 인덱스는 접두사 목록에 포함된 JSON 키만 인덱싱합니다. 인덱스의 키스페이스 접두사 목록 내에서 지정된 유형이 없는 키는 무시되며 검색 작업에 영향을 주지 않습니다.

HASH 또는 JSON 명령으로 인덱스의 키스페이스 내에 있는 키를 수정하면 해당 인덱스가 업데이트됩니다. 이 프로세스에는 각 인덱스에 대해 선언된 필드를 추출하고 새 값으로 인덱스를 업데이트하는 작업이 포함됩니다. 업데이트 프로세스는 백그라운드 스레드에서 수행되므로 인덱스가 최종적으로 해당 키스페이스 콘텐츠와 일치해야 합니다. 따라서 키를 삽입하거나 업데이트해도 당장은 검색 결과에 표시되지 않습니다. 시스템 부하가 심하거나 데이터가 심하게 변경되는 기간에는 가시성 지연이 더 길어질 수 있습니다.

인덱스 생성은 다단계 프로세스입니다. 첫 번째 단계는 인덱스를 정의하는 FT.CREATE 명령을 실행하는 것입니다. 생성을 성공적으로 실행하면 두 번째 단계인 채우기가 자동으로 시작됩니다. 채우기 프로세스는 백그라운드 스레드에서 실행되며 키스페이스를 스캔하여 새 인덱스의 접두사 목록 내에 있는 키를 찾습니다. 발견된 각 키는 인덱스에 추가됩니다. 결국 전체 키스페이스가 스캔되어 인덱스 생성 프로세스가 완료됩니다. 채우기 프로세스가 실행되는 동안에는 제한 없이 인덱스 키를 변경할 수 있으며 모든 키가 제대로 인덱싱될 때까지 인덱스 채우기 프로세스가 완료되지 않습니다. 인덱스를 채우는 동안 시도한 쿼리 작업은 허용되지 않으며 오류가 발생하여 종료됩니다. 채우기 프로세스의 완료 여부는 해당 인덱스에 대한 `FT.INFO` 명령 출력('backfill_status')에서 확인할 수 있습니다.

인덱스 필드 유형

인덱스의 각 필드(열)에는 인덱스 생성 시 선언되는 특정 유형 및 키 내의 위치가 표시됩니다. HASH 키의 경우 위치는 HASH 내의 필드 이름입니다. JSON 키의 경우 위치는 JSON 경로 설명입니다. 키가 수정되면 선언된 필드와 관련된 데이터가 추출되고 선언된 유형으로 변환되어 인덱스에 저장됩니다. 데이터가 누락되었거나 선언된 유형으로 성공적으로 변환할 수 없는 경우 해당 필드는 인덱스에서 제외됩니다. 필드에는 다음 설명과 같이 네 가지 유형이 있습니다.

- 숫자 필드에는 단일 숫자가 포함됩니다. JSON 필드의 경우 JSON 번호의 숫자 규칙을 따라야 합니다. HASH의 경우 필드에 고정 또는 부동 소수점 숫자의 표준 형식으로 작성된 ASCII 텍스트가 포함되어야 합니다. 키 내 표시 형식에 관계없이 이 필드는 64비트 부동 소수점 숫자로 변환되어 인덱스 내에 저장됩니다. 숫자 필드는 범위 검색 연산자와 함께 사용할 수 있습니다. 기본 숫자는 정밀도 제한에 따라 부동 소수점으로 저장되므로 부동 소수점 숫자 비교에 대한 일반적인 규칙이 적용됩니다.
- 태그 필드에는 단일 UTF-8 문자열로 코딩된 0개 이상의 태그 값이 포함됩니다. 문자열은 선행 및 후행 공백이 제거된 구분 문자(기본값은 쉼표이지만 재정의할 수 있음)를 사용하여 태그 값으로 구문 분석됩니다. 단일 태그 필드에 원하는 수의 태그 값을 포함할 수 있습니다. 태그 필드를 사용하여 대소문자를 구분하거나 구분하지 않는 비교를 통해 태그 값이 동일한 쿼리를 필터링할 수 있습니다.
- 텍스트 필드에는 UTF-8 규격을 준수하지 않아도 되는 수많은 바이트가 포함됩니다. 텍스트 필드를 사용하여 쿼리 결과를 애플리케이션에 적합한 값으로 장식할 수 있습니다. 예를 들어 URL이나 문서 내용 등이 가능합니다.
- 벡터 필드에는 임베딩이라고도 하는 숫자 벡터가 포함됩니다. 벡터 필드는 지정된 알고리즘과 거리 측정법을 사용하여 크기가 고정된 벡터의 K-nearest neighbor(KNN) 검색을 지원합니다. HASH 인덱스의 경우 필드에 바이너리 형식(리틀 엔디안 IEEE 754)으로 인코딩된 전체 벡터가 포함되어야 합니다. JSON 키의 경우 경로는 숫자로 채워진 올바른 크기의 배열을 참조해야 합니다. JSON 배열을 벡터 필드로 사용하는 경우 JSON 키 내 배열의 내부 표현이 선택한 알고리즘에서 요구하는 형식으로 변환되므로 메모리 사용량과 정밀도가 줄어듭니다. 이후에 JSON 명령을 사용하여 읽기 작업을 수행하면 낮아진 정밀도 값을 출력합니다.

벡터 인덱스 알고리즘

다음과 같은 두 개의 벡터 인덱스 알고리즘이 제공됩니다.

- FLAT - FLAT 알고리즘은 인덱스의 각 벡터를 무차별 대입으로 선형 처리하여 거리 계산의 정밀도 범위 내에서 정확한 답을 산출합니다. 인덱스의 선형 처리로 인해 큰 인덱스의 경우 이 알고리즘의 실행 시간이 매우 길어질 수 있습니다.

- Hierarchical Navigable Small Worlds(HSSW) - HNSW 알고리즘은 실행 시간을 크게 줄이는 대신 정답의 근사치를 제공합니다. 알고리즘은 세 개의 파라미터 M, EF_CONSTRUCTION, EF_RUNTIME에 의해 제어됩니다. 처음 두 파라미터는 인덱스 생성 시 지정되며 변경할 수 없습니다. 이 EF_RUNTIME 파라미터는 인덱스 생성 시 지정되는 기본값을 갖지만 이후에 개별 쿼리 작업에서 재정의할 수 있습니다. 이 세 파라미터는 상호 작용하여 수집 및 쿼리 작업 중에 메모리와 CPU 사용량의 균형을 맞추고 정확한 KNN 검색의 근사치 품질(재현율)을 제어합니다.

벡터 검색 알고리즘(FLAT 및 HNSW) 모두 선택적 INITIAL_CAP 파라미터를 지원합니다. 이 파라미터를 지정하면 인덱스에 메모리를 사전 할당하여 메모리 관리 오버헤드를 줄이고 벡터 수집 비율을 높입니다.

HNSW와 같은 벡터 검색 알고리즘은 이전에 삽입한 벡터의 삭제 또는 덮어쓰기를 효율적으로 처리하지 못할 수 있습니다. 이러한 연산을 사용하면 인덱스 메모리가 과도하게 소모되거나 재현율이 저하될 수 있습니다. 인덱스를 다시 지정하는 것은 최적의 메모리 사용량 또는 재현율을 복원하기 위한 방법 중 하나입니다.

벡터 검색 쿼리 표현식

FT.SEARCH 및 FT.AGGREGATE 명령에는 쿼리 표현식이 필요합니다. 이 표현식은 하나 이상의 연산자로 구성된 단일 문자열 파라미터입니다. 각 연산자는 인덱스의 한 필드를 사용하여 인덱스에 있는 키의 하위 집합을 식별합니다. 부울 컴바이너와 괄호로 여러 연산자를 결합하여 수집된 키 세트(또는 결과 집합)를 더욱 향상시키거나 제한할 수 있습니다.

와일드카드

와일드카드 연산자인 별표('*')는 인덱스의 모든 키와 일치합니다.

숫자 범위

숫자 범위 연산자의 구문은 다음과 같습니다.

```
<range-search> ::= '@' <numeric-field-name> ':' '[' <bound> <bound> ']'
<bound>  ::= <number> | '(' <number>
<number> ::= <integer> | <fixed-point> | <floating-point> | 'Inf' | '-Inf' | '+Inf'
```

<numeric-field-name>은 NUMERIC 유형의 선언된 필드여야 합니다. 기본적으로 경계값도 검색 범위에 포함되지만 앞에 여는 괄호['(']를 사용하여 경계값이 검색 범위에서 제외되도록 만들 수 있습니다. 범위 검색은 Inf, +Inf 또는 -Inf를 경계값으로 사용하여 단일 관계형 비교(<, <=, >, >=)로 변환될 수 있

습니다. 지정된 숫자 형식(정수, 고정 소수점, 부동 소수점, 무한대)에 관계없이 숫자를 64비트 부동 소수점으로 변환하여 비교를 수행하므로 정밀도가 낮아집니다.

Example 예시

```
@numeric-field:[0 10]                       // 0   <= <value> <= 10
@numeric-field:[(0 10]                      // 0   <  <value> <= 10
@numeric-field:[0 (10]                      // 0   <= <value> <  10
@numeric-field:[(0 (10]                     // 0   <  <value> <  10
@numeric-field:[1.5 (Inf]                   // 1.5 <= value
```

태그 비교

태그 비교 연산자의 구문은 다음과 같습니다.

```
<tag-search> ::= '@' <tag-field-name> ':' '{' <tag> [ '|' <tag> ]* '}'
```

연산자의 태그 중 하나라도 레코드 태그 필드에 있는 태그와 일치하는 경우 해당 레코드가 결과 집합에 포함됩니다. <tag-field-name>에서 디자인하는 필드는 TAG 유형으로 선언된 인덱스의 필드여야 합니다. 태그 비교의 예는 다음과 같습니다.

```
@tag-field:{ atag }
@tag-field: { tag1 | tag2 }
```

부울 조합

부울 논리 and/or를 사용하여 숫자 연산자 또는 태그 연산자의 결과 집합을 결합할 수 있습니다. 괄호를 사용하여 연산자를 그룹화하거나 평가 순서를 변경할 수 있습니다. 부울 논리 연산자의 구문은 다음과 같습니다.

```
<expression> ::= <phrase> | <phrase> '|' <expression> | '(' <expression> ')'
<phrase> ::= <term> | <term> <phrase>
<term> ::= <range-search> | <tag-search> | '*'
```

여러 용어가 하나의 구문으로 결합될 때 ‘and’ 연산이 수행됩니다. 여러 구문이 파이프(‘|’)로 결합될 때 'or' 연산이 수행됩니다.

벡터 검색

벡터 인덱스는 가장 가까운 이웃과 범위의 두 가지 검색 방법을 지원합니다. 가장 가까운 이웃 검색은 인덱스에서 제공된 (참조) 벡터에 가장 가까운 벡터의 숫자 K를 찾습니다. 이를 'K'개의 가장 가까운 이웃을 의미하는 KNN이라고 합니다. KNN 검색의 구문은 다음과 같습니다.

```
<vector-knn-search> ::= <expression> '=>[KNN' <k> '@' <vector-field-name> '$'
 <parameter-name> <modifiers> ']'
<modifiers> ::= [ 'EF_RUNTIME' <integer> ] [ 'AS' <distance-field-name>]
```

벡터 KNN 검색은 와일드카드, 범위 검색, 태그 검색 및/또는 이들 연산자의 부울 조합 등 위에서 정의된 연산자의 조합인 `<expression>`을 만족하는 벡터에만 적용됩니다.

- `<k>`는 반환되는 가장 가까운 이웃 벡터의 수를 지정하는 정수입니다.
- `<vector-field-name>`은 선언된 `VECTOR` 유형의 필드를 지정해야 합니다.
- `<parameter-name>` 필드는 `FT.SEARCH` 또는 `FT.AGGREGATE` 명령의 `PARAM` 테이블 항목 중 하나를 지정합니다. 이 파라미터는 거리 계산을 위한 참조 벡터 값입니다. 벡터 값은 리틀 엔디안 IEEE 754 바이너리 형식의 `PARAM` 값으로 인코딩됨(HASH 벡터 필드와 동일하게 인코딩됨)
- HNSW 유형의 벡터 인덱스의 경우 선택적 `EF_RUNTIME`절을 사용하여 인덱스 생성 시 설정된 `EF_RUNTIME` 파라미터의 기본값을 재정의할 수 있습니다.
- 선택적 `<distance-field-name>`은 결과 집합에 참조 벡터와 찾은 키 사이의 계산된 거리를 포함하는 필드 이름을 제공합니다.

범위 검색은 참조 벡터에서 지정된 거리(반경) 내에 있는 모든 벡터를 찾습니다. 범위 검색의 구문은 다음과 같습니다.

```
<vector-range-search> ::= ‘@’ <vector-field-name> ‘:’ ‘[’ ‘VECTOR_RANGE’ ( <radius> |
 ‘$’ <radius-parameter> )  $<reference-vector-parameter> ‘]’ [ ‘=’ ‘>’ ‘{’ <modifiers>
 ‘}’ ]
<modifiers> ::= <modifier> | <modifiers>, <modifier>
<modifer> ::= [ ‘$yield_distance_as’ ‘:’ <distance-field-name> ] [ ‘$epsilon’ ‘:’
 <epsilon-value> ]
```

위치:

- `<vector-field-name>`은 검색할 벡터 필드의 이름입니다.
- `<radius> or $<radius-parameter>`는 검색의 숫자 거리 제한입니다.

- `$<reference-vector-parameter>` 는 참조 벡터가 포함된 파라미터의 이름입니다. 벡터 값은 리틀 엔디안 IEEE 754 바이너리 형식의 PARAM 값으로 인코딩됩니다(HASH 벡터 필드와 동일하게 인코딩됨).
- 선택적 `<distance-field-name>`은 결과 집합에 참조 벡터와 각 키 사이의 계산된 거리를 포함하는 필드 이름을 제공합니다.
- 선택적 `<epsilon-value>` 는 검색 작업의 경계를 제어합니다. `<radius> * (1.0 + <epsilon-value>)` 거리 내의 벡터는 후보 결과를 찾기 위해 순회합니다. 기본값은 0.01입니다.

INFO 명령

벡터 검색은 통계 및 카운터의 여러 추가 섹션으로 Valkey 및 Redis OSS INFO 명령을 보강합니다. `SEARCH` 섹션 검색을 요청하면 다음 섹션이 모두 검색됩니다.

`search_memory` 섹션

명칭	설명
search_used_memory_bytes	모든 검색 데이터 구조에서 사용된 메모리 바이트 수
search_used_memory_human	사람이 읽을 수 있는 위의 버전

`search_index_stats` 섹션

명칭	설명
search_number_of_indexes	생성된 인덱스 수
search_num_fulltext_indexes	모든 인덱스의 벡터가 아닌 필드 수
search_num_vector_indexes	모든 인덱스 내 벡터 필드 수
search_num_hash_indexes	HASH 유형 키의 인덱스 수
search_num_json_indexes	JSON 유형 키의 인덱스 수
search_total_indexed_keys	모든 인덱스의 총 키 수

명칭	설명
search_total_indexed_vectors	모든 인덱스의 총 벡터 수
search_total_indexed_hash_keys	모든 인덱스의 HASH 유형 총 키 수
search_total_indexed_json_keys	모든 인덱스의 JASON 유형 총 키 수
search_total_index_size	모든 인덱스에서 사용하는 바이트
search_total_fulltext_index_size	벡터가 아닌 인덱스 구조에서 사용되는 바이트
search_total_vector_index_size	벡터 인덱스 구조에서 사용되는 바이트
search_max_index_lag_ms	마지막 수집 배치 업데이트 동안의 수집 지연

`search_ingestion` 섹션

명칭	설명
search_background_indexing_status	수집 상태 `NO_ACTIVITY` 는 유휴 상태를 의미합니다. 다른 값은 키가 수집 중인 키가 있음을 나타냅니다.
search_ingestion_paused	다시 시작하는 경우를 제외하고는 항상 'no'여야 합니다.

`search_backfill` 섹션

> **Note**
> 이 섹션에 설명된 일부 필드는 채우기가 진행 중일 때만 표시됩니다.

명칭	설명
search_num_active_backfills	현재 채우기 활동 수

명칭	설명
search_backfills_paused	메모리가 부족한 경우를 제외하고는 항상 'no'여야 합니다.
search_current_backfill_progress_percentage	현재 채우기 완료율(0~100)

`search_query` 섹션

명칭	설명
search_num_active_queries	현재 진행 중인 `FT.SEARCH` 명령 및 `FT.AGGREGATE` 명령 수

벡터 검색 보안

명령 및 데이터 액세스 모두에 대한 액세스 제어 목록(ACL) 보안 메커니즘이 검색 기능을 제어하도록 확장되었습니다. 개별 검색 명령의 ACL 제어가 완벽하게 지원됩니다. 새 ACL 범주 `@search`가 추가되었으며 많은 기존 범주(`@fast`, `@read`, `@write` 등)가 새 명령을 포함하도록 업데이트되었습니다. 검색 명령은 키 데이터를 수정하지 않습니다. 즉, 쓰기 액세스를 위한 기존 ACL 시스템이 보존됩니다. HASH 및 JSON 작업에 대한 액세스 규칙은 인덱스가 있더라도 수정되지 않습니다. 이러한 명령에는 여전히 일반적인 키 수준 액세스 제어가 적용됩니다.

인덱스가 있는 검색 명령의 액세스도 ACL로 제어됩니다. 액세스 확인은 키별 수준이 아닌 전체 인덱스 수준에서 수행됩니다. 즉, 사용자가 해당 인덱스의 키스페이스 접두사 목록 내에서 가능한 모든 키에 액세스할 수 있는 권한을 가진 경우에만 사용자에게 인덱스 액세스 권한이 부여됩니다. 즉, 인덱스의 실제 콘텐츠는 액세스를 제어하지 않습니다. 그보다는 접두사 목록에 정의된 인덱스의 이론상 내용이 보안 검사에 사용됩니다. 사용자에게 키에 대한 읽기 및/또는 쓰기 액세스는 있지만 해당 키가 포함된 인덱스에는 액세스할 수 없는 상황이 발생하기 쉽습니다. 인덱스를 만들거나 사용할 때는 키스페이스에 대한 읽기 액세스만 필요하며 쓰기 액세스 유무는 고려되지 않습니다.

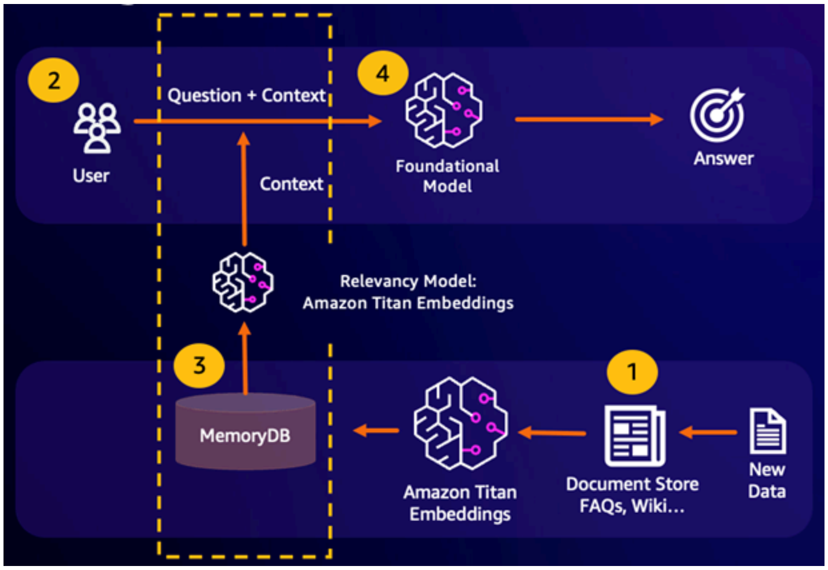
MemoryDB와 함께 ACL을 사용하는 방법을 자세히 알아보려면 액세스 제어 목록(ACL)을 사용하여 사용자 인증을 참조하세요.

사용 사례

벡터 검색의 사용 사례는 다음과 같습니다.

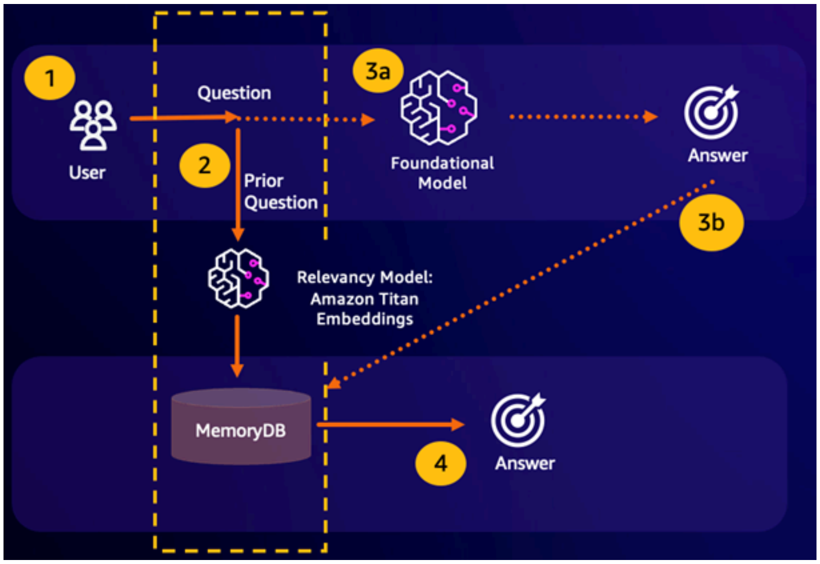
검색 증강 세대(RAG)

검색 증강 세대(RAG)는 벡터 검색으로 대규모 데이터에서 관련 구절을 검색하여 대규모 언어 모델(LLM)을 확장합니다. 특히, 인코더는 입력 컨텍스트와 검색 쿼리를 벡터에 임베딩한 다음 가장 가까운 이웃 검색을 사용하여 의미가 유사한 구절을 찾습니다. 이렇게 검색된 구절을 원본 컨텍스트와 연결되면 LLM에 추가적인 정보를 제공하여 사용자에게 보다 정확한 응답을 반환합니다.



내구성 있는 시맨틱 캐시

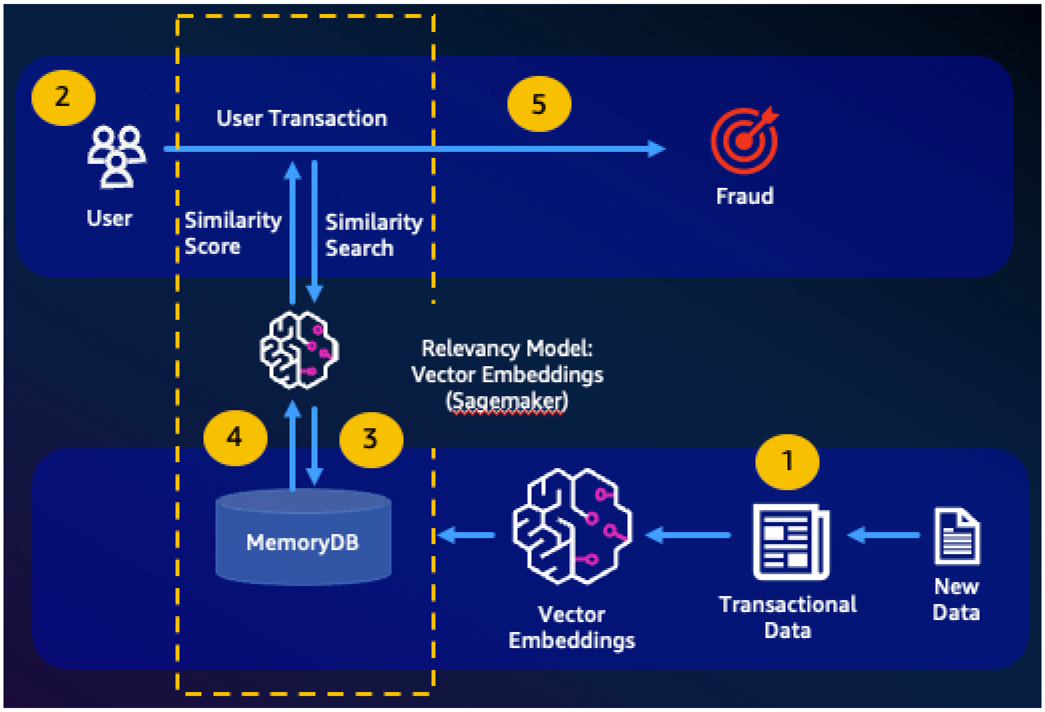
시맨틱 캐싱은 FM의 이전 결과를 저장하여 계산 비용을 줄이는 프로세스입니다. 시맨틱 캐싱은 이전 추론에서 얻은 이전 결과를 다시 계산하는 대신 재사용함으로써 FM을 통해 추론하는 데 필요한 계산량을 줄입니다. MemoryDB를 사용하면 내구성이 뛰어난 시맨팅 캐싱이 가능하므로 과거 추론의 데이터 손실을 방지할 수 있습니다. 이를 통해 생성형 AI 애플리케이션이 의미론적으로 유사한 이전 질문의 답변을 활용하여 한 자릿수 밀리초 내에 응답할 수 있으며 불필요한 LLM 추론이 방지되므로 비용을 절감할 수 있습니다.



- 시맨틱 검색 적중 - 정의된 유사성 점수를 기준으로 고객의 질의가 이전 질문과 의미상 유사한 경우 FM 버퍼 메모리(MemoryDB)는 4단계에서 이전 질문에 대한 답변을 반환하고 3단계에서는 FM을 호출하지 않습니다. 이렇게 하면 파운데이션 모델(FM)의 지연과 비용 발생을 방지할 수 있어 고객에게 더 빠른 답변을 제공할 수 있습니다.
- 시맨틱 검색 실패 - 정의된 유사성 점수를 기준으로 고객의 쿼리가 이전 쿼리와 의미상 유사하지 않은 경우 3a단계에서 FM을 호출하여 고객에게 답변을 제공합니다. 그런 다음 FM에서 생성된 응답은 향후 쿼리(3b단계)를 위해 MemoryDB에 벡터로 저장되어 의미상 유사한 질문에 대한 FM 비용을 최소화합니다. 이 흐름에서는 원래 쿼리에 의미상 유사한 질문이 없었기 때문에 4단계를 호출하지 않았습니다.

사기 탐지

이상 탐지의 한 형태인 사기 탐지는 유효한 거래를 벡터로 표현하고 완전히 새로운 신규 거래의 벡터 표현을 비교합니다. 완전히 새로운 신규 거래가 유효한 거래 데이터를 나타내는 벡터와 유사성이 낮을 때 사기 탐지가 이루어집니다. 이를 통해 가능한 모든 사기 인스턴스를 예측하는 대신 정상적인 행동을 모델링하여 사기를 탐지할 수 있습니다. MemoryDB를 사용하면 조직에서 오탐을 최소화하고 한 자릿수 밀리초의 지연 시간으로 처리량이 높은 기간에 이 작업을 수행할 수 있습니다.



그 외 사용 사례

- 추천 엔진은 항목을 벡터로 표현하여 사용자에게 유사한 제품이나 콘텐츠를 찾을 수 있습니다. 벡터는 속성과 패턴을 분석하여 만들어집니다. 사용자 패턴 및 속성을 기반으로 사용자로부터 좋은 평가를 받은 가장 유사한 벡터를 찾아 이전에 보지 못한 새로운 항목을 사용자에게 추천할 수 있습니다.
- 문서 검색 엔진은 텍스트 문서를 의미론적 의미를 포착하는 고밀도 숫자 벡터로 표현합니다. 검색 시 엔진은 검색 쿼리를 벡터로 변환하고 가장 가까운 이웃 검색을 사용하여 쿼리와 벡터가 가장 유사한 문서를 찾습니다. 이 벡터 유사성 접근 방식을 사용하면 단순히 키워드를 일치시키는 대신 의미를 기반으로 문서를 일치시킬 수 있습니다.

벡터 검색 기능 및 제한

벡터 검색 가능 여부

벡터 검색 활성화 MemoryDB 구성은 R6g, R7g 및 T4g 노드 유형에서 지원되며 MemoryDB를 사용할 수 있는 모든 AWS 리전에서 사용할 수 있습니다.

검색을 활성화하기 위해 기존 클러스터를 수정할 수 없습니다. 그러나 검색이 비활성화된 클러스터의 스냅샷에서 검색이 활성화된 클러스터를 생성할 수 있습니다.

파라미터 제한 사항

다음 표는 다양한 벡터 검색 항목에 대한 제한 사항을 보여줍니다.

아이템	최대값
벡터의 차원 수	32768
만들 수 있는 인덱스 수	10
인덱스의 필드 수	50
FT.SEARCH 및 FT.AGGREGATE TIMEOUT 절 (밀리초)	10000
FT.AGGREGATE 명령의 파이프라인 단계 수	32
FT.AGGREGATE LOAD 절의 필드 수	1024
FT.AGGREGATE GROUPBY 절의 필드 수	16
FT.AGGREGATE SORTBY 절의 필드 수	16
FT.AGGREGATE PARAM 절의 파라미터 수	32
HNSW M 파라미터	512
HNSW EF_CONSTRUCTION 파라미터	4096
HNSW EF_RUNTIME 파라미터	4096

규모 조정 제한

MemoryDB에 대한 벡터 검색은 현재 단일 샤드로 제한되며 수평적 크기 조정은 지원되지 않습니다. 벡터 검색은 수직 및 복제 크기 조정을 지원합니다.

운영상의 제한 사항

인덱스 지속성 및 채우기

벡터 검색 기능은 인덱스의 정의와 인덱스의 콘텐츠를 유지합니다. 즉, 노드를 시작하거나 다시 시작하게 하는 모든 운영 요청 또는 이벤트 중에 인덱스 정의 및 콘텐츠가 최신 스냅샷에서 복원되고 보류 중인 모든 트랜잭션이 저널에서 재생됩니다. 이를 시작하는 데는 사용자 작업이 필요하지 않습니다. 재구축은 데이터가 복원되는 즉시 채우기 작업으로 수행됩니다. 이는 시스템이 정의된 각 인덱스에 대해 FT.CREATE 명령을 자동으로 실행하는 것과 기능적으로 동일합니다. 데이터가 복원되자마자 노드를 애플리케이션 작업에 사용할 수 있지만 인덱스 채우기가 완료되지 않았을 가능성이 높습니다. 즉, 채우기가 애플리케이션에 다시 표시될 수 있으며, 예를 들어 인덱스 채우기를 사용한 검색 명령은 거부될 수 있습니다. 채우기에 대한 자세한 내용은 벡터 검색 개요을(를) 참조하세요.

인덱스 채우기 완료는 원본과 복제본 간에 동기화되지 않습니다. 불완전한 동기화가 애플리케이션에 예기치 않게 나타날 수 있으므로 애플리케이션에서 검색 작업을 시작하기 전에 원본과 모든 복제본에서 채우기가 완료되었는지 확인하는 것이 좋습니다.

스냅샷 가져오기/내보내기 및 실시간 마이그레이션

RDB 파일에 검색 인덱스가 있으면 해당 데이터 전송의 호환성이 제한됩니다. MemoryDB 벡터 검색 기능에 의해 정의된 벡터 인덱스의 형식은 다른 MemoryDB 벡터 활성화 클러스터에서만 인식됩니다. 또한 MemoryDB 클러스터의 GA 버전에서 평가판 클러스터의 RDB 파일을 가져올 수 있습니다. 그러면 RDB 파일을 로드할 때 인덱스 콘텐츠가 재구축됩니다.

그러나 인덱스가 포함되지 않은 RDB 파일에는 이러한 제한이 적용되지 않습니다. 따라서 내보내기 전에 인덱스를 삭제하여 평가판 클러스터 내의 데이터를 미리 보기가 아닌 클러스터로 내보낼 수 있습니다.

메모리 사용

메모리 소비는 벡터 수, 차원 수, M-값, 벡터가 아닌 데이터(예: 벡터에 연결된 메타데이터 또는 인스턴스에 저장된 기타 데이터)의 양을 기반으로 합니다.

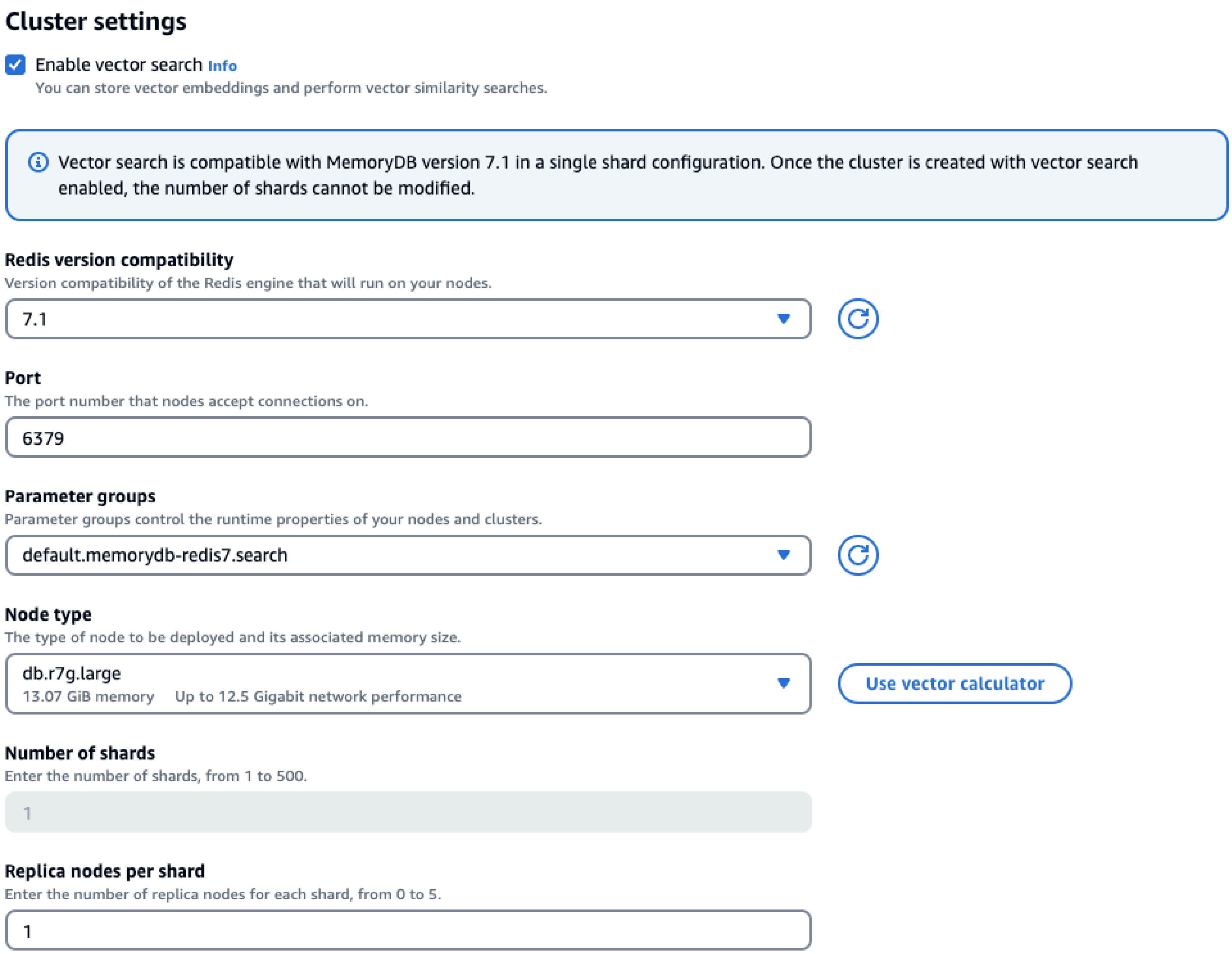
필요한 총 메모리는 실제 벡터 데이터에 필요한 공간과 벡터 인덱스에 필요한 공간의 조합입니다. 벡터 데이터에 필요한 공간은 최적의 메모리 할당을 위해 HASH 또는 JSON 데이터 구조 내에 벡터를 저장하는 데 필요한 실제 용량과 가장 가까운 메모리 슬래브에 대한 오버헤드를 측정하여 계산됩니다. 각 벡터 인덱스는 이러한 데이터 구조에 저장된 벡터 데이터에 대한 참조를 사용하며 효율적인 메모리 최적화를 사용하여 인덱스에 있는 벡터 데이터의 중복 사본을 제거합니다.

벡터 수는 데이터를 벡터로 표현하는 방법에 따라 달라집니다. 예를 들어 단일 문서를 여러 청크로 나타내도록 선택할 수 있습니다. 여기서 각 청크는 벡터를 나타냅니다. 또는 전체 문서를 단일 벡터로 나타내도록 선택할 수 있습니다.

벡터의 차원 수는 선택한 임베딩 모델에 따라 달라집니다. 예를 들어 AWS Titan 임베딩 모델을 사용하도록 선택하면 차원 수는 1536이 됩니다.

M 파라미터는 인덱스 구성 중에 모든 새 요소에 대해 생성된 양방향 링크 수를 나타냅니다. MemoryDB는 이 값을 16으로 기본 설정합니다. 하지만 이를 재정의할 수 있습니다. M 파라미터가 높으면 높은 차원성 및/또는 높은 재현율 요구 사항에 더 적합하고, M 파라미터가 낮으면 낮은 차원성 및/또는 낮은 재현율 요구 사항에 더 적합합니다. M 값은 인덱스가 커질수록 메모리 소비가 증가합니다.

MemoryDB 콘솔 환경에서 클러스터 설정의 벡터 검색 활성화를 선택하면 벡터 워크로드의 특성에 따라 올바른 인스턴스 유형을 쉽게 선택할 수 있습니다.

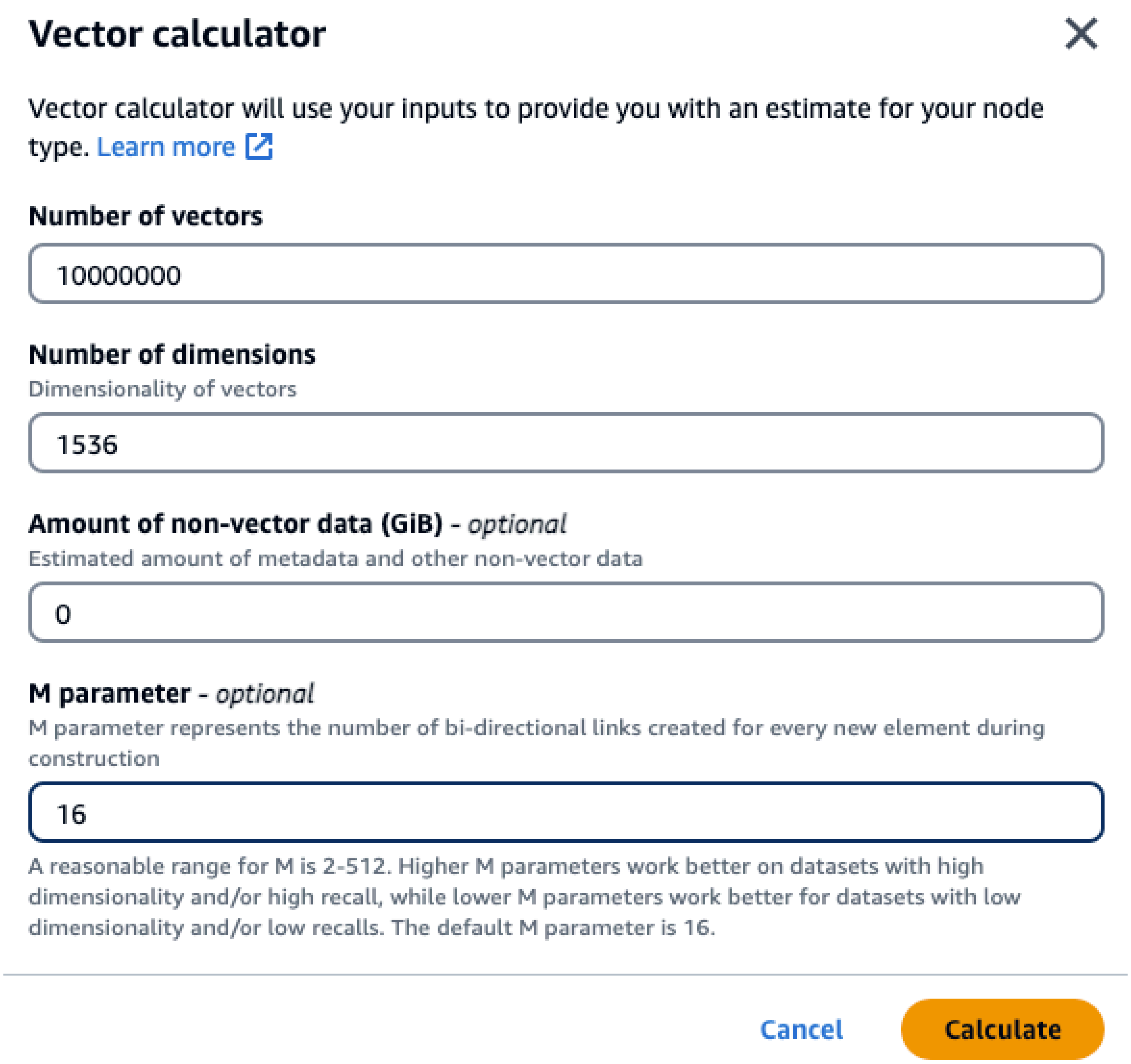


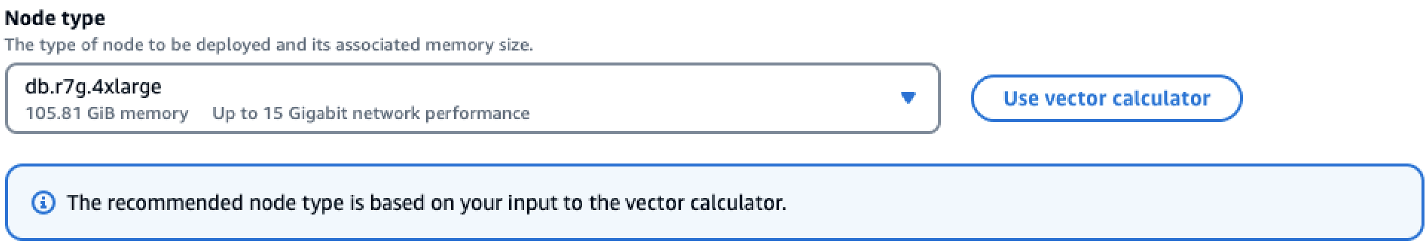
샘플 워크로드

고객이 내부 재무 문서를 기반으로 시맨틱 검색 엔진을 구축하려고 합니다. 고객은 현재 1536개 차원을 갖는 titan 임베딩 모델을 사용하여 문서당 10개의 벡터로 청크된 100만 개 재무 문서를 보유하고 있으며 벡터가 아닌 데이터는 없습니다. 고객은 기본값인 16을 M 파라미터로 사용하기로 결정합니다.

- 벡터: 100만 * 10 청크 = 1000만 벡터
- 차원: 1536
- 벡터가 아닌 데이터(GB): 0GB
- M 파라미터: 16

고객은 이 데이터를 사용하고 콘솔에서 벡터 계산기 사용 버튼을 클릭하여 파라미터를 기반으로 권장 인스턴스 유형을 가져올 수 있습니다.





이 예제에서 벡터 계산기는 제공된 파라미터를 기반으로 벡터를 저장하는 데 필요한 메모리를 보유할 수 있는 가장 작은 MemoryDB r7g 노드 유형을 찾습니다. 이는 근사치이며, 인스턴스 유형을 테스트하여 요구 사항에 맞는지 확인해야 합니다.

위의 계산 방법과 샘플 워크로드의 파라미터에 따라 이 벡터 데이터는 데이터와 단일 인덱스를 저장하는 데 104.9GB가 필요합니다. 이 경우 사용 가능한 메모리가 105.81GB인 `db.r7g.4xlarge` 인스턴스 유형이 권장됩니다. 다음으로 가장 작은 노드 유형은 너무 작아서 벡터 워크로드를 보유할 수 없습니다.

각 벡터 인덱스는 저장된 벡터 데이터에 대한 참조를 사용하고 벡터 인덱스에 벡터 데이터의 추가 사본을 생성하지 않으므로 인덱스는 비교적 적은 공간을 소비합니다. 이는 여러 인덱스를 만드는 경우뿐만 아니라, 벡터 데이터의 일부가 삭제되어 고품질 벡터 검색 결과를 위한 최적의 노드 연결을 만들도록 HNSW 그래프를 재구성하는 경우에도 매우 유용합니다.

채우기 중 메모리 부족

Valkey 및 Redis OSS 쓰기 작업과 마찬가지로 인덱스 채우기에는 메모리 부족 제한이 적용됩니다. 채우기가 진행되는 동안 엔진 메모리가 가득 차면 모든 채우기가 일시 중지됩니다. 메모리를 사용할 수 있게 되면 채우기가 다시 시작됩니다. 메모리 부족으로 인해 채우기가 일시 중지된 경우 삭제하고 인덱싱할 수도 있습니다.

트랜잭션

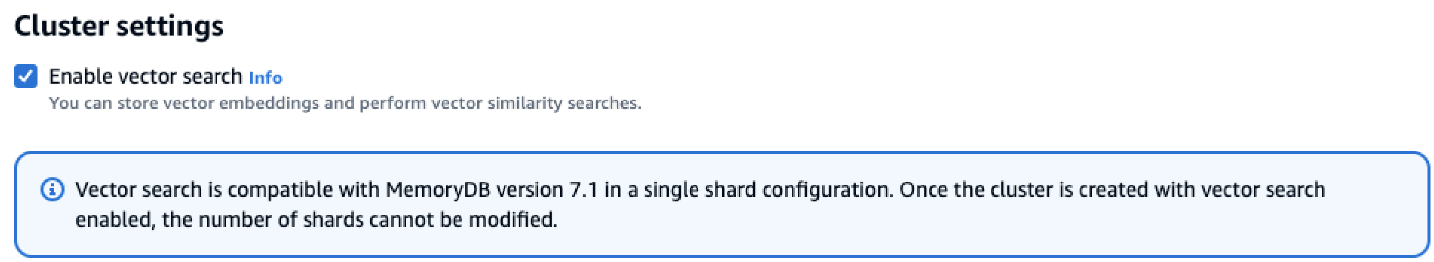
`FT.CREATE`, `FT.DROPINDEX`, `FT.ALIASADD`, `FT.ALIASDEL`, 및 `FT.ALIASUPDATE` 명령은 트랜잭션 컨텍스트에서 실행할 수 없습니다. 즉, MULTI/EXEC 블록, LUA 또는 FUNCTION 스크립트 내에서는 실행할 수 없습니다.

벡터 검색이 가능한 클러스터 생성

AWS Management 콘솔 또는 AWS Command Line Interface를 사용하여 벡터 검색이 가능한 클러스터를 생성할 수 있습니다. 접근 방식에 따라 벡터 검색을 활성화하기 위한 고려 사항을 구현해야 합니다.

AWS Management 콘솔 사용

콘솔 내에서 벡터 검색이 가능한 클러스터를 만들려면 클러스터 설정에서 벡터 검색을 활성화해야 합니다. 벡터 검색은 단일 샤드 구성의 MemoryDB 버전 7.1에서 사용할 수 있습니다.



AWS Management 콘솔에서 벡터 검색을 사용하는 방법에 대한 자세한 내용은 클러스터 생성(콘솔)을(를) 참조하세요.

AWS Command Line Interface 사용

벡터 검색이 가능한 MemoryDB 클러스터를 생성하려면 MemoryDB create-cluster 명령을 사용하여 변경 불가능한 파라미터 그룹 `default.memorydb-redis7.search`를 전달하여 벡터 검색 기능을 활성화하면 됩니다.

```
aws memorydb create-cluster \
  --cluster-name <value> \
  --node-type <value> \
  --engine redis \
  --engine-version 7.1 \
  --num-shards 1 \
  --acl-name <value> \
  --parameter-group-name default.memorydb-redis7.search
```

선택적으로 다음 예제와 같이 새 파라미터 그룹을 생성하여 벡터 검색을 활성화할 수도 있습니다. 파라미터 그룹에 대한 자세한 내용은 여기에서 확인할 수 있습니다.

```
aws memorydb create-parameter-group \
  --parameter-group-name my-search-parameter-group \
  --family memorydb_redis7
```

그런 다음 새로 생성된 파라미터 그룹에서 search-enabled 파라미터를 yes로 업데이트합니다.

```
aws memorydb update-parameter-group \
```

```
--parameter-group-name my-search-parameter-group \
--parameter-name-values "ParameterName=search-enabled,ParameterValue=yes"
```

이제 기본 파라미터 그룹 대신 이 사용자 지정 파라미터 그룹을 사용하여 MemoryDB 클러스터에서 벡터 검색을 활성화할 수 있습니다.

벡터 검색 명령

다음은 벡터 검색에 지원되는 명령 목록입니다.

주제

- FT.CREATE
- FT.SEARCH
- FT.AGGREGATE
- FT.DROPINDEX
- FT.INFO
- FT._LIST
- FT.ALIASADD
- FT.ALIASDEL
- FT.ALIASUPDATE
- FT._ALIASLIST
- FT.PROFILE
- FT.EXPLAIN
- FT.EXPLAINCLI

FT.CREATE

인덱스를 생성하고 해당 인덱스 채우기를 시작합니다. 인덱스 구성에 대한 자세한 내용은 벡터 검색 개요를 참조하세요.

구문

```
FT.CREATE <index-name>
ON HASH | JSON
[PREFIX <count> <prefix1> [<prefix2>...]]
```

```
SCHEMA
(<field-identifier> [AS <alias>]
  NUMERIC
| TAG [SEPARATOR <sep>] [CASESENSITIVE]
| TEXT
| VECTOR [HNSW|FLAT] <attr_count> [<attribute_name> <attribute_value>])

)+
```

스키마

- 필드 식별자:
  - HASH 키의 경우 필드 식별자는 필드 이름입니다.
  - JSON 키의 경우 필드 식별자는 JSON 경로입니다.

  자세한 내용은 인덱스 필드 유형 섹션을 참조하세요.

- 필드 유형:
  - TAG: 자세한 내용은 태그를 참조하세요.
  - NUMERIC: 필드에는 숫자가 포함됩니다.
  - TEXT: 필드에는 모든 데이터 BLOB이 포함됩니다.
  - VECTOR: 벡터 검색을 지원하는 벡터 필드입니다.
    - 알고리즘 - Hierarchical Navigable Small Worlds(HSNW) 또는 무차별 암호 대입(FLAT)일 수 있습니다.
    - `attr_count` - 알고리즘 구성으로 전달되는 속성의 수입니다. 여기에는 이름과 값이 모두 포함됩니다.
    - `{attribute_name} {attribute_value}` - 인덱스 구성을 정의하는 알고리즘별 키/값 쌍입니다.

      FLAT 알고리즘의 경우 속성은 다음과 같습니다.

      필수 항목 여부:

      - DIM - 벡터의 차원 개수입니다.
      - DISTANCE_METRIC - [L2 | IP | COSINE] 중 하나일 수 있습니다.
      - TYPE - 벡터 유형입니다. `FLOAT32` 유형만 지원됩니다.

      선택 사항:

- INITIAL_CAP - 인덱스의 초기 벡터 용량은 인덱스의 메모리 할당 크기에 영향을 줍니다.

HNSW 알고리즘의 경우 속성은 다음과 같습니다.

필수 항목 여부:

- TYPE - 벡터 유형입니다. `FLOAT32` 유형만 지원됩니다.
- DIM - 벡터 차원으로, 양의 정수로 지정됩니다. 최댓값: 32,768
- DISTANCE_METRIC - [L2 | IP | COSINE] 중 하나일 수 있습니다.

선택 사항:

- INITIAL_CAP - 인덱스의 초기 벡터 용량은 인덱스의 메모리 할당 크기에 영향을 줍니다. 기본값은 1024입니다.
- M - 각 계층의 그래프에서 각 노드에 허용되는 최대 발신 엣지 수입니다. 계층 0에서 최대 발신 엣지 수는 2M입니다. 기본값은 16이고 최대값은 512입니다.
- EF_CONSTRUCTION - 인덱스 생성 중에 검사되는 벡터 수를 제어합니다. 이 파라미터의 값이 높을수록 인덱스 생성 시간이 길어지는 대신 재현율이 향상됩니다. 기본값은 200입니다. 최대값은 4,096입니다.
- EF_RUNTIME - 쿼리 작업 중에 검사되는 벡터 수를 제어합니다. 이 파라미터의 값이 높을수록 쿼리 시간이 길어지는 대신 재현율이 향상될 수 있습니다. 이 파라미터의 값은 쿼리별로 재정의할 수 있습니다. 기본값은 10입니다. 최대값은 4,096입니다.

반환

간단한 문자열 OK 메시지 또는 오류 응답을 반환합니다.

예시

> **Note**
>
> 다음 예시에서는 Valkey 또는 Redis OSS로 전송하기 전에 valkey-cli 고유의 인수(예: 데이터 인용 제거 및 이스케이프 제거)를 사용합니다. 다른 프로그래밍 언어 클라이언트(Python, Ruby, C# 등)를 사용하려면 해당 환경의 문자열 및 이진 데이터 처리 규칙을 따릅니다. 자세한 내용은 AWS 기반 구축 도구를 참조하세요.

Example 1: 인덱스 만들기

크기가 2인 벡터에 대한 인덱스 만들기

```
FT.CREATE hash_idx1 ON HASH PREFIX 1 hash: SCHEMA vec AS VEC VECTOR HNSW 6 DIM 2 TYPE
 FLOAT32 DISTANCE_METRIC L2
OK
```

HNSW 알고리즘을 사용하여 6차원 JSON 인덱스를 생성합니다.

```
FT.CREATE json_idx1 ON JSON PREFIX 1 json: SCHEMA $.vec AS VEC VECTOR HNSW 6 DIM 6 TYPE
 FLOAT32 DISTANCE_METRIC L2
OK
```

Example 예제 2: 일부 데이터 채우기

다음 명령은 redis-cli 터미널 프로그램에 대한 인수로 실행할 수 있도록 형식이 지정되었습니다. 프로그래밍 언어 클라이언트(Python, Ruby, C# 등)를 사용하는 개발자는 해당 환경의 문자열 및 이진 데이터 처리 규칙을 따라야 합니다.

일부 해시 및 json 데이터 생성:

```
HSET hash:0 vec "\x00\x00\x00\x00\x00\x00\x00\x00"
HSET hash:1 vec "\x00\x00\x00\x00\x00\x00\x80\xbf"
JSON.SET json:0 . '{"vec":[1,2,3,4,5,6]}'
JSON.SET json:1 . '{"vec":[10,20,30,40,50,60]}'
JSON.SET json:2 . '{"vec":[1.1,1.2,1.3,1.4,1.5,1.6]}'
```

다음 사항에 유의하세요.

- HASH 및 JSON 데이터의 키에는 인덱스 정의의 접두사가 있습니다.
- 벡터는 인덱스 정의의 적절한 경로에 있습니다.
- HASH 벡터는 16진수 데이터로 입력되고 JSON 데이터는 숫자로 입력됩니다.
- 벡터는 적절한 길이이고, 2차원 HASH 벡터 항목은 부동 소수점 2자리의 16진수 데이터로 구성되며, 6차원 JSON 벡터 항목은 6자리 숫자로 구성됩니다.

Example 예제 3: 인덱스 삭제 후 다시 생성

```
FT.DROPINDEX json_idx1
OK

FT.CREATE json_idx1 ON JSON PREFIX 1 json: SCHEMA $.vec AS VEC VECTOR FLAT 6 DIM 6 TYPE
 FLOAT32 DISTANCE_METRIC L2
```

```
OK
```

새 JSON 인덱스는 FLAT 알고리즘 대신 HNSW 알고리즘을 사용합니다. 또한 기존 JSON 데이터를 다시 인덱싱합니다.

```
FT.SEARCH json_idx1 "*=>[KNN 100 @VEC $query_vec]" PARAMS 2 query_vec
 "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
 DIALECT 2
1) (integer) 3
2) "json:2"
3) 1) "__VEC_score"
   2) "11.11"
   3) "$"
   4) "[{\"vec\":[1.1, 1.2, 1.3, 1.4, 1.5, 1.6]}]"
4) "json:0"
5) 1) "__VEC_score"
   2) "91"
   3) "$"
   4) "[{\"vec\":[1.0, 2.0, 3.0, 4.0, 5.0, 6.0]}]"
6) "json:1"
7) 1) "__VEC_score"
   2) "9100"
   3) "$"
   4) "[{\"vec\":[10.0, 20.0, 30.0, 40.0, 50.0, 60.0]}]"
```

FT.SEARCH

제공된 쿼리 표현식을 사용하여 인덱스 내에서 키를 찾습니다. 찾은 후에는 해당 키 내에 있는 인덱싱된 필드의 수 및/또는 내용을 반환할 수 있습니다. 자세한 내용은 벡터 검색 쿼리 표현식을 참조하세요.

이러한 예제에서 사용할 데이터를 생성하려면 FT.CREATE 명령을 참조하세요.

구문

```
FT.SEARCH <index-name> <query>
[RETURN <token_count> (<field-identifier> [AS <alias>])+]
[TIMEOUT timeout]
[PARAMS <count> <name> <value> [<name> <value>]]
[LIMIT <offset> <count>]
[COUNT]
```

- RETURN: 이 절은 반환되는 키 필드를 식별합니다. 각 필드의 선택적 AS 절은 결과의 필드 이름을 대체합니다. 이 인덱스에 대해 선언된 필드만 지정할 수 있습니다.
- LIMIT: <offset> <count>: 이 절은 오프셋 및 개수 값을 충족하는 키만 반환하는 페이지 매김 기능을 제공합니다. 이 절을 생략하면 기본값은 'LIMIT 0 10'입니다. 즉, 최대 10개의 키만 반환됩니다.
- PARAMS: 키 값 쌍 수의 2배입니다. 파라미터 키/값 쌍은 쿼리 표현식 내에서 참조할 수 있습니다. 자세한 내용은 벡터 검색 쿼리 표현식을 참조하세요.
- COUNT: 이 절은 키 콘텐츠 반환을 억제하고 키 수만 반환합니다. 이는 'LIMIT 0 0'의 별칭입니다.

반환

배열 또는 오류 응답을 반환합니다.

- 작업이 성공적으로 완료되면 배열을 반환합니다. 첫 번째 요소는 쿼리와 일치하는 총 키 수입니다. 나머지 요소는 키 이름과 필드 목록 쌍입니다. 필드 목록은 필드 이름과 값 쌍으로 구성된 또 다른 배열입니다.
- 인덱스가 다시 채워지고 있는 경우 명령은 즉시 오류 응답을 반환합니다.
- 제한 시간에 도달하면 명령이 오류 응답을 반환합니다.

예시: 몇 가지 검색 수행

> **Note**
>
> 다음 예시에서는 Valkey 또는 Redis OSS로 전송하기 전에 valkey-cli 고유의 인수(예: 데이터 인용 제거 및 이스케이프 제거)를 사용합니다. 다른 프로그래밍 언어 클라이언트(Python, Ruby, C# 등)를 사용하려면 해당 환경의 문자열 및 이진 데이터 처리 규칙을 따릅니다. 자세한 내용은 AWS 기반 구축 도구를 참조하세요.

해시 검색

```
FT.SEARCH hash_idx1 "*=>[KNN 2 @VEC $query_vec]" PARAMS 2 query_vec
 "\x00\x00\x00\x00\x00\x00\x00\x00" DIALECT 2
1) (integer) 2
2) "hash:0"
3) 1) "__VEC_score"
   2) "0"
   3) "vec"
```

```
   4) "\x00\x00\x00\x00\x00\x00\x00\x00"
4) "hash:1"
5) 1) "__VEC_score"
   2) "1"
   3) "vec"
   4) "\x00\x00\x00\x00\x00\x00\x80\xbf"
```

쿼리 벡터와의 거리(16진수)인 점수를 기준으로 정렬된 결과 두 개가 생성됩니다.

JSON 검색

```
FT.SEARCH json_idx1 "*=>[KNN 2 @VEC $query_vec]" PARAMS 2 query_vec
 "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
 DIALECT 2
1) (integer) 2
2) "json:2"
3) 1) "__VEC_score"
   2) "11.11"
   3) "$"
   4) "[{\"vec\":[1.1, 1.2, 1.3, 1.4, 1.5, 1.6]}]"
4) "json:0"
5) 1) "__VEC_score"
   2) "91"
   3) "$"
   4) "[{\"vec\":[1.0, 2.0, 3.0, 4.0, 5.0, 6.0]}]"
```

가장 근사한 결과가 두 개가 점수별로 정렬되어 생성됩니다. 참고로 JSON 벡터 값은 부동 수로 변환되고 쿼리 벡터는 벡터 데이터입니다. 또한 KNN 파라미터가 2이기 때문에 결과가 두 개만 반환됩니다. 값이 클수록 더 많은 결과가 반환됩니다.

```
FT.SEARCH json_idx1 "*=>[KNN 100 @VEC $query_vec]" PARAMS 2 query_vec
 "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
 DIALECT 2
1) (integer) 3
2) "json:2"
3) 1) "__VEC_score"
   2) "11.11"
   3) "$"
   4) "[{\"vec\":[1.1, 1.2, 1.3, 1.4, 1.5, 1.6]}]"
4) "json:0"
5) 1) "__VEC_score"
   2) "91"
```

```
    3) "$"
    4) "[{\"vec\":[1.0, 2.0, 3.0, 4.0, 5.0, 6.0]}]"
6) "json:1"
7) 1) "__VEC_score"
   2) "9100"
   3) "$"
   4) "[{\"vec\":[10.0, 20.0, 30.0, 40.0, 50.0, 60.0]}]"
```

FT.AGGREGATE

FT.SEARCH 명령의 상위 집합으로, 쿼리 표현식으로 선택한 키의 상당 수를 추가로 처리할 수 있습니다.

구문

```
FT.AGGREGATE index query
  [LOAD * | [count field [field ...]]]
  [TIMEOUT timeout]
  [PARAMS count name value [name value ...]]
  [FILTER expression]
  [LIMIT offset num]
  [GROUPBY count property [property ...] [REDUCE function count arg [arg ...] [AS name]
 [REDUCE function count arg [arg ...] [AS name] ...]] ...]]
  [SORTBY count [ property ASC | DESC [property ASC | DESC ...]] [MAX num]]
  [APPLY expression AS name]
```

- FILTER, LIMIT, GROUPBY, SORTBY 및 APPLY 절을 원하는 순서대로 여러 번 반복할 수 있으며 자유롭게 결합할 수 있습니다. 이는 지정된 순서대로 적용되며, 한 절의 출력이 다음 절의 입력이 됩니다.
- 위 구문에서 '속성'은 이 인덱스에 대해 FT.CREATE 명령으로 선언된 필드이거나 이전 APPLY 절 또는 REDUCE 함수의 출력입니다.
- LOAD 절은 인덱스에서 선언된 필드를 로드하는 것으로 제한됩니다. 'LOAD *'는 인덱스에 선언된 모든 필드를 로드합니다.
- 지원되는 리듀서 함수는 COUNT, COUNT_DISTINCTISH, SUM, MIN, MAX, AVG, STDDEV, QUANTILE, TOLIST, FIRST_VALUE, RANDOM_SAMPLE입니다. 자세한 내용은 집계를 참조하세요.
- LIMIT <offset> <count>: <offset>에서 시작하여 <count>로 이어지는 레코드를 유지하며, 다른 모든 레코드는 삭제됩니다.
- PARAMS: 키 값 쌍 수의 2배입니다. 파라미터 키/값 쌍은 쿼리 표현식 내에서 참조할 수 있습니다.

반환

배열 또는 오류 응답을 반환합니다.

- 작업이 성공적으로 완료되면 배열을 반환합니다. 첫 번째 요소는 특별한 의미가 없는 정수입니다(무시해야 함). 나머지 요소는 마지막 단계에서 출력된 결과입니다. 각 요소는 필드 이름 및 값 쌍의 배열입니다.
- 인덱스가 다시 채워지고 있는 경우 명령은 즉시 오류 응답을 반환합니다.
- 제한 시간에 도달하면 명령이 오류 응답을 반환합니다.

FT.DROPINDEX

인덱스를 삭제합니다. 인덱스 정의 및 관련 콘텐츠가 삭제됩니다. 키는 영향을 받지 않습니다.

구문

```
FT.DROPINDEX <index-name>
```

반환

간단한 문자열 OK 메시지 또는 오류 응답을 반환합니다.

FT.INFO

구문

```
FT.INFO <index-name>
```

FT.INFO 페이지의 출력은 다음 표에 설명된 대로 키 값 쌍의 배열입니다.

키	값 유형	설명
index_name	문자열	인덱스의 이름
creation_timestamp	정수	생성 시간의 유닉스 스타일 타임스탬프
key_type	문자열	해시 또는 JSON

키	값 유형	설명
key_prefixes	문자열 배열	이 인덱스의 키 접두사
필드	필드 정보 배열	이 인덱스의 필드
space_usage	정수	이 인덱스에서 사용하는 메모리 바이트
fullext_space_usage	정수	벡터가 아닌 필드에서 사용되는 메모리 바이트
vector_space_usage	정수	벡터 필드에서 사용되는 메모리 바이트
num_docs	정수	현재 인덱스에 포함된 키 수
num_indexed_vectors	정수	현재 인덱스에 포함된 벡터 수
current_lag	정수	최근 수집 지연(밀리초)
backfill_status	문자열	다음 중 하나: 완료, 진행 중, 일시 중지됨 또는 실패

다음 테이블은 각 필드에 대한 정보를 나타냅니다.

키	값 유형	설명
식별자	문자열	필드 이름
field_name	문자열	해시 멤버 이름 또는 JSON 경로
유형	문자열	숫자, 태그, 텍스트 또는 벡터 중 하나
옵션	문자열	무시

필드 유형이 Vector인 경우 알고리즘에 따라 추가 정보가 표시됩니다.

HNSW 알고리즘의 경우:

키	값 유형	설명
알고리즘	문자열	HNSW
data_type	문자열	FLOAT32
distance_metric	문자열	다음 중 하나: L2, IP 또는 코사인
initial_capacity	정수	벡터 필드 인덱스의 초기 크기
current_capacity	정수	현재 벡터 필드 인덱스 크기
maximum_edges	정수	생성 시 M 파라미터
ef_construction	정수	생성 시 EF_CONSTRUCTION 파라미터
ef_runtime	정수	생성 시 EF_RUNTIME 파라미터

FLAT 알고리즘의 경우:

키	값 유형	설명
알고리즘	문자열	FLAT
data_type	문자열	FLOAT32
distance_metric	문자열	다음 중 하나: L2, IP 또는 코사인
initial_capacity	정수	벡터 필드 인덱스의 초기 크기
current_capacity	정수	현재 벡터 필드 인덱스 크기

FT._LIST

모든 인덱스를 나열합니다.

구문

```
FT._LIST
```

반환

인덱스 이름의 배열 반환

FT.ALIASADD

색인에 별칭을 추가합니다. 새 별칭 이름은 인덱스 이름이 필요한 모든 곳에서 사용할 수 있습니다.

구문

```
FT.ALIASADD <alias> <index-name>
```

반환

간단한 문자열 OK 메시지 또는 오류 응답을 반환합니다.

FT.ALIASDEL

색인의 기존 별칭을 삭제합니다.

구문

```
FT.ALIASDEL <alias>
```

반환

간단한 문자열 OK 메시지 또는 오류 응답을 반환합니다.

FT.ALIASUPDATE

다른 물리적 인덱스를 가리키도록 기존 별칭을 업데이트합니다. 이 명령은 별칭에 대한 향후 참조에만 영향을 줍니다. 진행 중인 작업(FT.SEARCH, FT.AGGREGATE)은 이 명령의 영향을 받지 않습니다.

구문

```
FT.ALIASUPDATE <alias> <index>
```

반환

간단한 문자열 OK 메시지 또는 오류 응답을 반환합니다.

FT._ALIASLIST

인덱스 별칭을 나열합니다.

구문

```
FT._ALIASLIST
```

반환

현재 별칭 개수 크기의 배열을 반환합니다. 배열의 각 요소는 별칭-인덱스 쌍입니다.

FT.PROFILE

쿼리를 실행하고 해당 쿼리에 대한 프로필 정보를 반환합니다.

구문

```
FT.PROFILE

<index>
SEARCH | AGGREGATE
[LIMITED]
QUERY <query ....>
```

반환

요소를 2개 가진 배열입니다. 첫 번째 요소는 프로파일링된 `FT.SEARCH` 또는`FT.AGGREGATE` 명령의 결과입니다. 두 번째 요소는 성능 및 프로파일링 정보의 배열입니다.

FT.EXPLAIN

쿼리를 구문 분석하고 해당 쿼리가 어떻게 구문 분석되었는지에 대한 정보를 반환합니다.

구문

```
FT.EXPLAIN <index> <query>
```

반환

구문 분석된 결과를 포함하는 문자열입니다.

FT.EXPLAINCLI

결과가 다른 형식으로 표시된다는 점을 제외하면 FT.EXPLAIN 명령과 동일합니다. redis-cli를 사용할 때 더 유용합니다.

구문

```
FT.EXPLAINCLI <index> <query>
```

반환

구문 분석된 결과를 포함하는 문자열입니다.

MemoryDB Multi-Region

MemoryDB Multi-Region은 완전 관리형 액티브-액티브 다중 리전 데이터베이스로, 최대 99.999%의 가용성과 마이크로초 읽기 및 한 자릿수 밀리초 쓰기 지연 시간으로 다중 리전 애플리케이션을 구축할 수 있습니다. 지역적 성능 저하를 통해 가용성과 복원력을 모두 향상시키는 동시에 다중 리전 애플리케이션의 로컬 읽기 및 쓰기 지연 시간이 짧은 이점을 누릴 수 있습니다.

MemoryDB Multi-Region을 사용하면 복원력을 높이기 위해 가용성이 높은 다중 리전 애플리케이션을 구축할 수 있습니다. 액티브-액티브 복제를 제공하므로 마이크로초 읽기 및 한 자릿수 밀리초 쓰기 지연 시간으로 고객과 가장 가까운 리전에서 로컬로 읽기 및 쓰기를 제공할 수 있습니다. MemoryDB Multi-Region 은 리전 간에 데이터를 비동기적으로 복제하며 데이터는 일반적으로 1초 이내에 전파됩니다. 업데이트 충돌을 자동으로 해결하고 데이터 발산 문제를 수정하여 애플리케이션에 집중할 수 있습니다.

MemoryDB Multi-Region이 현재 지원되는 AWS 리전은 다음과 같습니다. 미국 동부(버지니아 북부 및 오하이오), 미국 서부(오리건, 캘리포니아 북부), 유럽(아일랜드, 프랑크푸르트, 런던), 아시아 태평양(도쿄, 시드니, 뭄바이, 서울, 싱가포르).

몇 번의 클릭만으로 AWS Management 콘솔 또는 최신 AWS SDK 또는 AWS CLI를 사용하여 MemoryDB Multi-Region을 쉽게 시작할 수 있습니다.

주제

사전 조건 및 제한 사항

MemoryDB Multi-Region을 시작하기 전에 다음 사항에 유의하세요.

- MemoryDB Multi-Region은 선택한 리전 간에 데이터를 복제합니다 - Multi-Region 클러스터를 생성하면 선택한 리전 간에 데이터가 이동된다는 것을 이해하고 이에 동의하는 것입니다.

  Multi-Region 그룹에서 리전을 제거하면 해당 리전의 리전 클러스터도 삭제됩니다.
- 리전별 가용성 - MemoryDB Multi-Region이 지원되는 AWS 리전은 다음과 같습니다. 미국 동부(버지니아 북부 및 오하이오), 미국 서부(오리건, 캘리포니아 북부), 유럽(아일랜드, 프랑크푸르트, 런던), 아시아 태평양(도쿄, 시드니, 뭄바이, 서울, 싱가포르).
- 동작 및 설정 - 모든 Multi-Region 리전 클러스터는 동일한 수의 샤드, 인스턴스 유형, Valkey 엔진 버전, TLS 및 파라미터 그룹 설정을 갖습니다. 각 리전 클러스터에 대해 서로 다른 IAM 인증, ACLs, 스냅샷 창, 태그, 고객 관리형 키(CMK) 및 유지 관리 기간을 선택할 수 있습니다.

  MemoryDB Multi-Region을 사용하면 다른 리전의 클러스터에 다른 수의 복제본이 있을 수 있습니다.
- 지원되는 노드 유형 - MemoryDB Multi-Region은 크기가 XL 이상인 R7g 노드에서 지원됩니다.

  MemoryDB Multi-Region은 Valkey 엔진 버전 7.3 이상을 지원합니다.
- 지원되는 데이터 유형 - MemoryDB Multi-Region은 현재 대부분의 Redis OSS 또는 Valkey 데이터 유형을 지원하며 향후 더 많은 데이터 유형에 대한 지원을 추가할 예정입니다. 지원되는 데이터 형식에는 문자열, 해시, 집합 및 정렬된 집합이 포함되지만 이러한 데이터 형식을 조작하는 모든 명령이 지원되는 것은 아닙니다.
- 총 리전 수 - MemoryDB Multi-Region을 사용하면 최대 5개의 AWS 리전 간에 MemoryDB 클러스터 데이터를 자동으로 복제할 수 있습니다.
- 지원되는 옵션 - MemoryDB Multi-Region은 수평/수직적 스케일링, IAM 통합, ACLs, 자동 및 온디맨드 스냅샷 생성, 자동 소프트웨어 패치 적용 및 모니터링을 지원합니다.
- 백업 및 복원 - 스냅샷을 생성하여 Multi-Region 리전 클러스터의 데이터를 백업할 수 있습니다. 스냅샷을 수동으로 생성하거나 MemoryDB의 자동 스냅샷 스케줄러를 사용하여 각 리전 클러스터에 대해 개별적으로 지정한 시간에 매일 새 스냅샷을 생성할 수 있습니다.
- 마이그레이션 - MemoryDB 또는 Redis OSS/Valkey RDB 형식 백업을 복원하도록 선택할 수 있습니다. 백업에서 데이터를 마이그레이션하려면 새 MemoryDB Multi-Region 리전 클러스터를 생성하고 Amazon S3에서 스냅샷 위치를 지정합니다. MemoryDB 스냅샷인 경우 이름을 지정할 수도 있습니다. MemoryDB Multi-Region은 스냅샷의 데이터를 사용하여 리전 클러스터를 생성합니다. MemoryDB Multi-Region은 문자열, 해시, 집합, 정렬된 집합 데이터 형식을 지원하므로 지원되는 이러한 데이터 유형에 대해서만 스냅샷 데이터를 마이그레이션할 수 있습니다. 백업 파일에 지원되지 않는 Redis OSS 데이터 형식이 포함된 경우 MemoryDB Multi-Region은 기본적으로 마이그레이션 작업에 실패합니다.
- 리소스 예약 - MemoryDB Multi-Region은 리전 가용성을 보호하도록 설계되었습니다. 일부 리소스는 피어 리전의 워크로드와 독립적으로 로컬 읽기 및 쓰기 요청을 제공할 수 있도록 각 노드에 영구적으

로 예약됩니다. 또한 이러한 리소스는 리전 격리 이벤트 및 복구를 포함하여 피어 리전의 이벤트 중에 로컬 가용성을 보호하는 역할을 합니다. 이로 인해 단일 리전 MemoryDB와 비교하여 성능 특성이 달라집니다. MemoryDB Multi-Region은 수평 및 수직적 스케일링을 모두 지원하여 사용 가능한 리소스를 확장합니다.

- RPO/RTO SLAs 없음 - MemoryDB Multi-Region은 명시된 RPO/RTO SLA를 제공하지 않습니다. 다른 AWS 리전에서 격리된 AWS 리전의 쓰기는 계속 수락되므로 교차 복제 지연이 무기한 증가할 수 있습니다. 고객은 “MultiRegionClusterReplicationLag” 지표를 사용하여 격리를 감지하고 원하는 RPO에 따라 애플리케이션 트래픽을 다른 리전으로 리디렉션해야 합니다.
- 단일 엔드포인트 또는 자동 장애 조치 없음: - 리전 중단 시 고객 트래픽을 다른 리전의 애플리케이션 스택으로 수동으로 리디렉션해야 합니다. MemoryDB 클러스터에 대한 다중 리전 액세스를 올바르게 구성했는지 확인해야 합니다.
- TTL 지원 없음 - MemoryDB Multi-Region은 TTL(Time to Live)을 지원하지 않습니다.
- 데이터 계층화 또는 벡터 검색 지원 없음 - MemoryDB Multi-Region은 벡터 검색 및 데이터 계층화 기능을 지원하지 않습니다.
- MemoryDB Multi-Region은 read-modify-write 명령(APPEND, RENAMENX 등)을 지원하지 않습니다.
- MemoryDB Multi-Region에서는 Redis OSS 트랜잭션 원자성과 일관성이 보장되지 않습니다.
- 인증 모델 - 지원되는 모든 리전에서 MemoryDB Multi-Region API 작업을 호출할 수 있습니다. IAM 정책에서 다중 리전 클러스터의 ARN을 지정하여 권한 범위를 제한할 수 있습니다. 다중 리전 클러스터 ARN의 형식은 `arn:aws:memorydb::<account-id>:multiregioncluster/multi-region-cluster-name`입니다. ARN에는 리전 정보가 없습니다.
- 처리량 제한 - MemoryDB Multi-Region은 리전의 노드당 최대 1.3GB/s의 읽기 처리량과 샤드당 최대 50MB/s의 전역 집계 쓰기 처리량을 지원할 수 있습니다.
- AWS 정책 - AWS ReadOnlyAccess 정책은 AWS 서비스 및 리소스에 대한 읽기 전용 액세스를 제공하지만 하나 이상의 다중 리전 클러스터에 대한 세부 정보를 자동으로 검색하지는 않습니다. 하나 이상의 다중 리전 클러스터에 대한 세부 정보를 검색하려면 AmazonMemoryDBReadOnlyAccess 정책을 사용하거나 IAM 고객 관리형 정책을 생성합니다.
- 리전 클러스터 삭제 - 리전 클러스터를 삭제할 때 연결된 고객 관리형 키(CMK)는 리전 클러스터 삭제가 완료될 때까지 유효해야 합니다. 이렇게 하면 나머지 리전 클러스터가 일관된 상태로 수렴될 수 있습니다.

작동 방법

MemoryDB Multi-Region의 작동 방식은 다음과 같습니다.

- 개념:

  Multi-Region 클러스터는 하나 이상의 리전 클러스터 모음으로, 모두 단일 AWS 계정에서 소유합니다.

  리전 클러스터는 Multi-Region 클러스터의 일부인 AWS 리전의 단일 클러스터입니다. 각 리전 클러스터는 동일한 데이터 세트를 저장합니다. 지정된 Multi-Region 클러스터에는 AWS 리전당 하나의 리전 클러스터만 있을 수 있습니다.

  Multi-Region 클러스터를 생성할 때 MemoryDB가 단일 단위로 취급하는 여러 리전 클러스터(리전당 하나)로 구성됩니다. 애플리케이션이 리전 클러스터에 데이터를 쓰면 MemoryDB는 해당 데이터를 Multi-Region 클러스터 내의 다른 모든 리전 클러스터에 자동으로 비동기식으로 복제합니다. 리전 클러스터를 Multi-Region 클러스터에 추가하여 추가 리전에서 사용할 수 있도록 할 수 있습니다. 최대 5개의 리전 간에 MemoryDB 클러스터 데이터를 자동으로 복제할 수 있습니다.

- 가용성과 내구성

  드물지만 리전의 리전 격리 또는 성능 저하가 발생할 경우 데이터베이스 재구성 없이 트래픽을 다른 정상 리전 중 하나로 리디렉션하도록 글로벌 DNS를 업데이트하여 애플리케이션의 고가용성을 유지하는 프로세스를 간소화할 수 있습니다. MemoryDB는 모든 리전의 모든 쓰기를 다중 AZ 트랜잭션 로그에 내구성 있게 저장하여 리전 내에서 데이터가 손실되지 않도록 합니다. MemoryDB Multi-Region은 리전에서 승인되었지만 아직 모든 멤버 클러스터에 복제되지 않은 모든 쓰기를 추적합니다. 리전이 격리되거나 성능이 저하되는 경우에도 로컬 쓰기는 계속 수락됩니다. 격리된 리전이 Multi-Region 클러스터에 다시 연결되면 승인되었지만 아직 다른 리전에 복제되지 않은 쓰기는 Multi-Region 클러스터의 모든 리전에 복제됩니다. 또한 MemoryDB Multi-Region은 CRDT 메커니즘을 사용하여 중단 중에 다른 리전에서 발생했을 수 있는 업데이트와 보류 중인 쓰기를 자동으로 조정합니다.

- MemoryDB Multi-Region 클러스터에 연결

  리전 클러스터에 데이터를 쓰고 리전 클러스터에서 데이터를 읽으려면 지원되는 Redis OSS/Valkey 클라이언트(Valkey GLIDE 포함) 중 하나를 사용하여 연결합니다. 각 리전 클러스터에는 Redis OSS/Valkey 클라이언트가 연결할 수 있는 엔드포인트가 있습니다. AWS 콘솔, CLI 또는 API를 사용하여 리전 클러스터 엔드포인트를 검색할 수 있습니다. 그런 다음 애플리케이션에서 이 엔드포인트를 사용(또는 구성)하여 리전 클러스터에서 데이터를 읽거나 쓸 수 있습니다.

정합성 및 충돌 해결

리전 클러스터 중 하나의 키에 대한 모든 업데이트는 일반적으로 1초 이내에 Multi-Region 클러스터의 다른 리전 클러스터에 비동기적으로 전파됩니다. 리전이 분리되거나 저하되면 MemoryDB Multi-Region은 수행된 쓰기 기록을 유지하지만 모든 복제본 테이블로 전파하지는 않습니다. 리전이 다시 온라인 상태로 되면 MemoryDB Multi-Region은 해당 리전에서 보류 중인 쓰기 전파를 재개하여 다른 리전의 멤버 클러스터로 전파합니다. 다시 온라인 상태로 된 리전에 대한 다른 멤버 클러스터의 쓰기 전파도 재개됩니다. 이전에 성공한 모든 쓰기 작업은 리전이 격리된 기간에 관계없이 결국 전파됩니다.

애플리케이션 업데이트가 서로 다른 리전에서 동시에 동일한 키를 업데이트할 경우 충돌이 발생할 수 있습니다. MemoryDB Multi-Region은 충돌 없는 복제 데이터 형식(CRDT)을 사용하여 충돌하는 동시 쓰기 간에 조정합니다. CRDT는 조정 없이 독립적으로 동시에 업데이트할 수 있는 데이터 구조입니다. 즉, 쓰기-쓰기 충돌이 최종 일관성을 갖추고 각 복제본에서 독립적으로 병합됩니다.

세부 정보에서 MemoryDB는 2가지 수준의 최종 라이터 우선 적용(LWW)을 사용하여 충돌을 해결합니다. 문자열 데이터 유형의 경우 LWW는 키 수준에서 충돌을 해결합니다. 다른 데이터 유형의 경우 LWW는 하위 키 수준에서 충돌을 해결합니다. 충돌 해결은 완벽하게 관리되며 애플리케이션 가용성에 영향을 주지 않고 백그라운드에서 이루어집니다. 다음은 해시 데이터 형식의 예입니다.

리전 A는 타임스탬프 T1에서 “HSET K F1 V1”을 실행하고, 리전 B는 타임스탬프 T2에서 “HSET K F2 V2”를 실행합니다. 복제 후 리전 A와 B 모두 두 필드가 있는 키 K를 갖습니다. 서로 다른 리전이 동일한 컬렉션에서 서로 다른 하위 키를 동시에 업데이트하는 경우 MemoryDB는 해시 데이터 유형에 대해 하위 키 수준에서 충돌을 해결하기 때문에 두 업데이트가 서로 충돌하지 않습니다. 따라서 최종 데이터에는 두 업데이트의 효과가 포함됩니다.

시간	리전 A	리전 B
T1	HSET K F1 V1	
T2		HSET K F2 V2
T3	동기화	동기화
T4	K: {F1:V1, F2:V2}	K: {F1:V1, F2:V2}

CRDT 및 예제

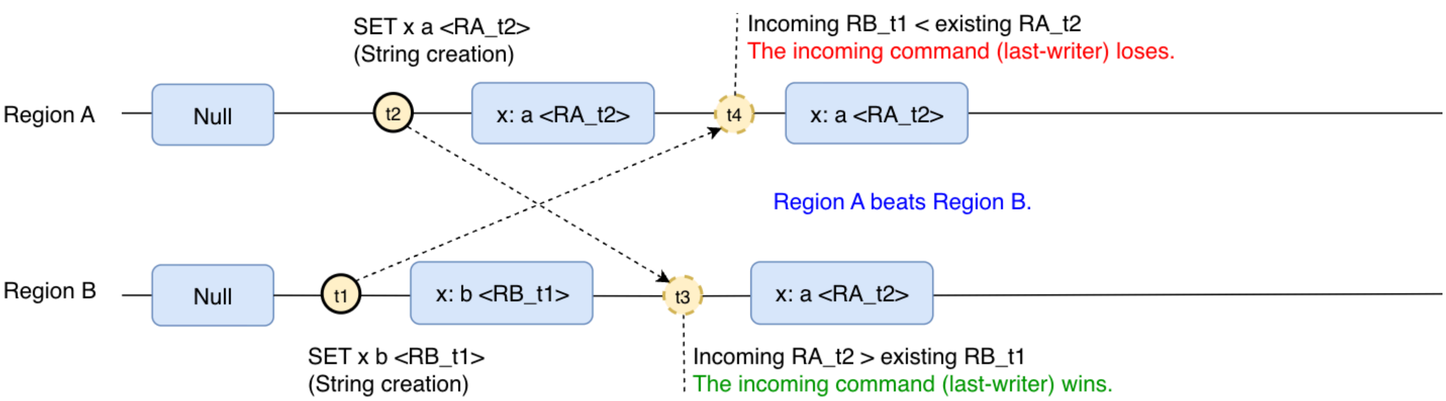
MemoryDB Multi-Region은 충돌 없는 복제 데이터 형식(CRDT)을 구현하여 여러 리전에서 발생한 동시 쓰기 충돌을 해결합니다. CRDT를 사용하면 순서에 관계없이 동일한 작업 세트를 최종적으로 수신한 후 서로 다른 리전에서 최종 일관성을 독립적으로 달성할 수 있습니다.

단일 키가 여러 리전에서 동시에 업데이트되는 경우 데이터 일관성을 달성하려면 쓰기 충돌을 해결해야 합니다. MemoryDB Multi-Region은 최종 라이터 우선 적용(LWW) 전략을 사용하여 성공 작업을 결정하고 "이후에 발생"하는 작업의 영향만 최종적으로 관찰합니다. op2가 실행될 때 op1의 효과가 Regionit에 원래 실행된 경우 작업 op1이 작업 op2에 "이전에 발생"했다고 가정합니다.

컬렉션(Hash, Set, SortedSet)의 경우 MemoryDB Multi-Region은 요소 수준에서 충돌을 해결합니다. 이를 통해 MemoryDB Multi-Region은 LWW를 사용하여 각 요소의 쓰기/쓰기 충돌을 해결할 수 있습니다. 예를 들어 여러 리전의 동일한 컬렉션에 다른 요소를 동시에 추가하면 컬렉션에 모든 요소가 포함됩니다.

동시 실행: 최종 라이터 우선 적용

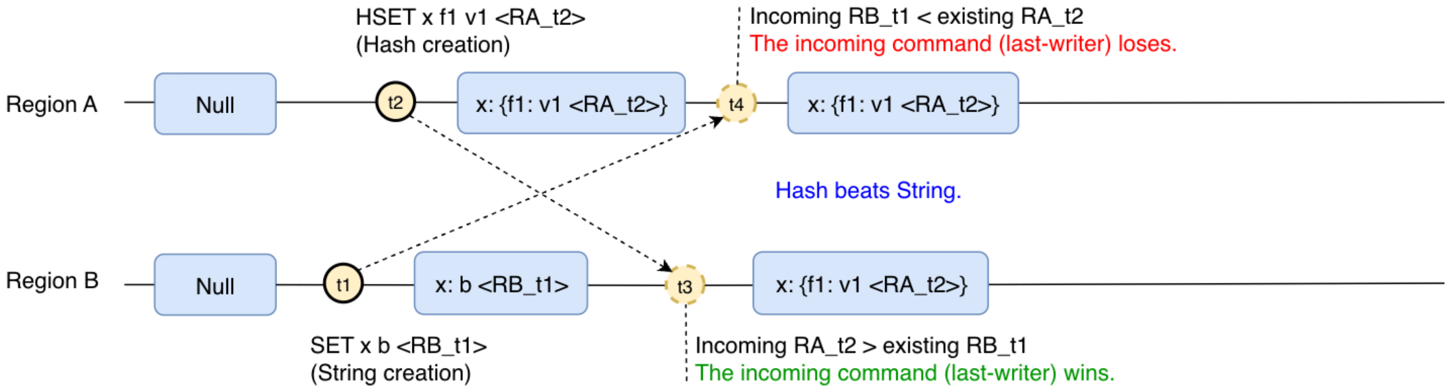
MemoryDB Multi-Region에서 키를 동시에 생성할 때 모든 리전에서 실행된 마지막 작업이 키의 결과를 결정합니다. 예:



키 x는 리전 B에서 값이 "b"로 생성되었지만 그 이후에는 리전 A에서 동일한 키가 값이 "a"로 생성되었습니다. 리전 A의 작업이 마지막으로 수행된 작업이므로 결국 키는 "a" 값을 갖도록 수렴됩니다.

데이터 형식이 충돌하는 동시 실행: 최종 라이터 우선 적용

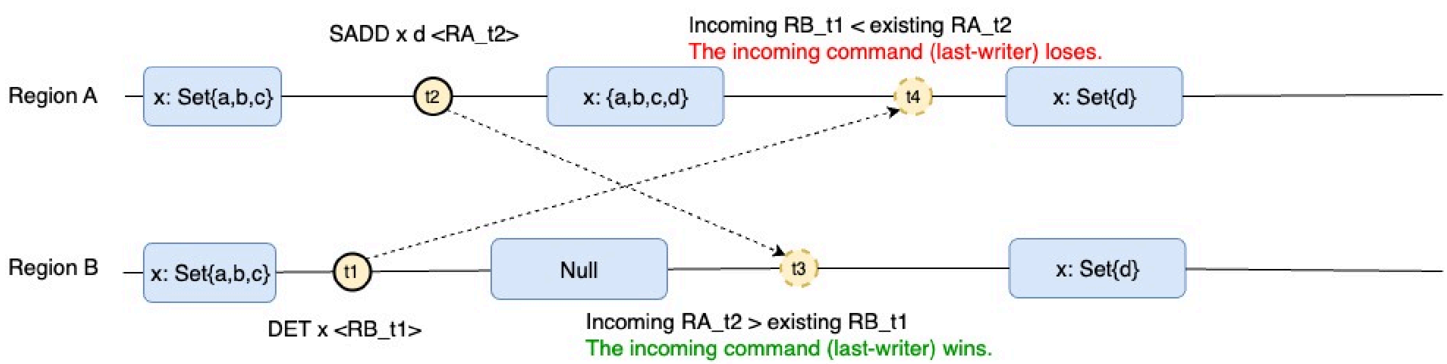
이전 예제에서 키는 두 리전 모두에서 동일한 유형으로 생성되었습니다. 키가 다른 데이터 유형으로 생성된 경우에도 유사한 동작을 볼 수 있습니다.



키 x는 리전 B에서 값이 "b"인 String으로 생성되었습니다. 하지만 그 이후에는 해당 작업이 리전 A에 복제되기 전에 리전 A에 Hash로 동일한 키가 생성됩니다. 리전 A의 작업이 마지막으로 수행된 작업이므로 결국 키가 리전 A에서 생성된 Hash로 수렴됩니다.

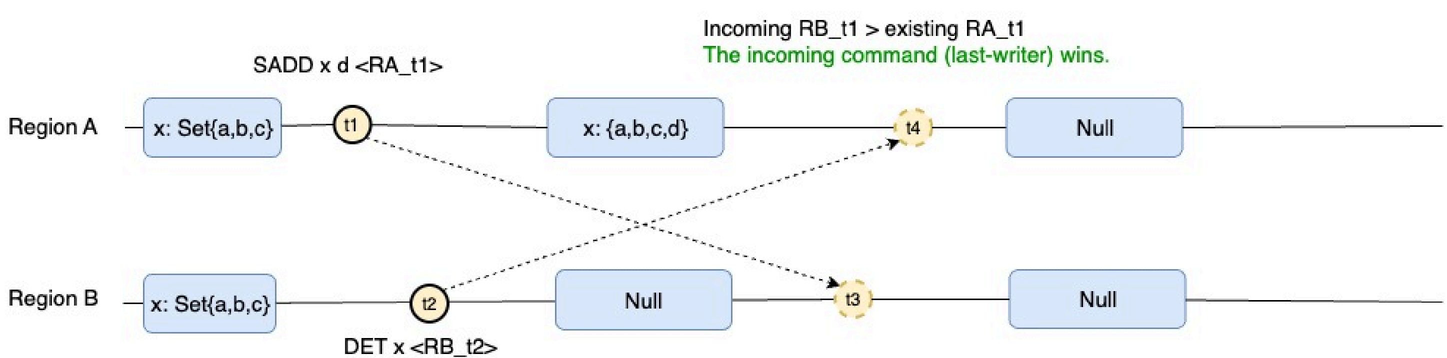
동시 생성-삭제: 최종 라이터 우선 적용

동시 삭제 및 “생성”(값 대체/추가)이 있는 시나리오에서는 마지막으로 수행된 작업이 우선 적용됩니다. 최종 결과는 삭제 작업의 순서에 따라 결정됩니다. 삭제가 이전에 발생한 경우:



리전 B에서 세트 유형의 키 x가 삭제되었습니다. 그 후 리전 A에서 해당 키에 새 멤버가 추가되었습니다. 리전 A의 작업이 마지막으로 수행된 작업이므로 결국 키는 리전 A에 추가된 유일한 요소가 있는 세트로 수렴됩니다.

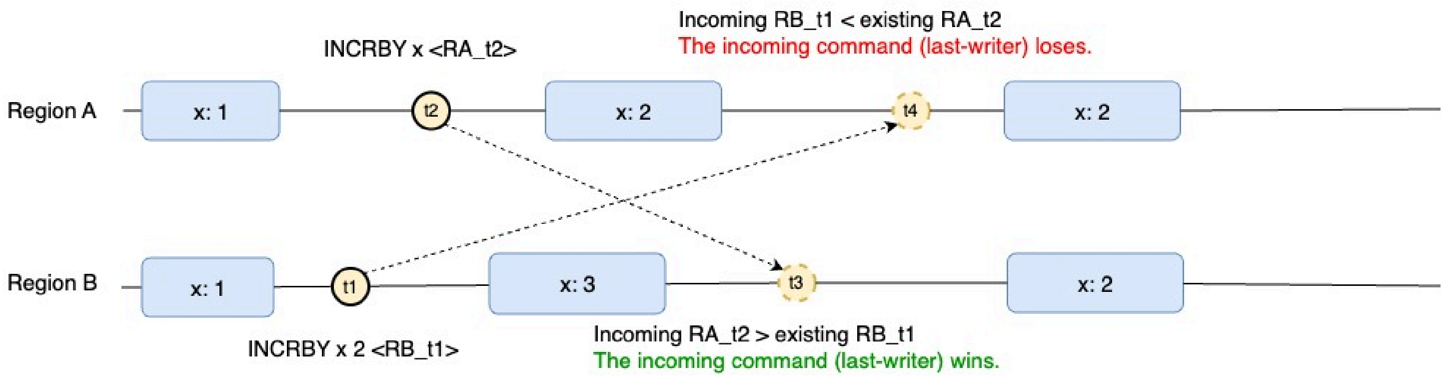
삭제가 나중에 발생한 경우:



리전 A에 설정된 유형의 키 x에 새 멤버가 추가되었습니다. 이후 리전 B에서 키가 삭제되었습니다. 리전 B에 대한 작업이 마지막으로 수행된 작업이므로 결국 키가 삭제되도록 수렴됩니다.

카운터, 동시 작업: 최종 라이터 우선 적용으로 전체 값 복제

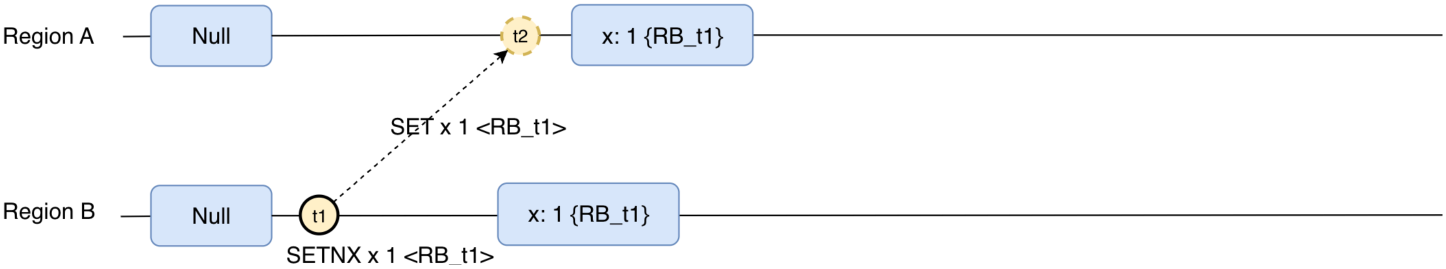
MemoryDB Multi-Region의 카운터는 전체 값 복제를 수행하고 최종 라이터 전략을 적용하여 비카운터 유형과 유사하게 작동합니다. 동시 작업은 결합되지 않지만 마지막 작업은 대신 성공합니다. 예:



이 시나리오에서 키 x의 시작 값은 1입니다. 그런 다음 리전 B는 카운터 x를 2만큼 늘린 다음 곧바로 리전 A는 카운터를 1만큼 늘렸습니다. 리전 A가 마지막으로 수행된 작업이므로 마지막으로 수행된 작업이 1만큼 증가임에 따라 키 x는 결국 값 2로 수렴됩니다.

비결정적 명령은 결정적 명령으로 복제

여러 리전에서 값의 일관성을 보장하기 위해 MemoryDB Multi-Region에서 비결정적 명령은 결정적 명령으로 복제됩니다. 비결정적 명령은 SETNX와 같은 외부 요인에 의존하는 명령입니다. SETNX는 키의 존재 여부에 따라 달라지며 키는 원격 리전에는 존재하지만 명령을 수신하는 로컬 리전에는 존재하지 않을 수 있습니다. 따라서 비결정적 명령은 전체 값 복제로 복제됩니다. 문자열의 경우 SET 명령으로 복제됩니다.



요약하면 문자열 유형에 대한 모든 작업은 SET 또는 DEL로 복제되고, 해시 유형에 대한 모든 작업은 HSET 또는 HDEL로 복제되며, 설정 유형에 대한 모든 작업은 SADD 또는 SREM으로 복제되고, 정렬된 세트에 대한 모든 작업은 ZADD 또는 ZREM으로 복제됩니다.

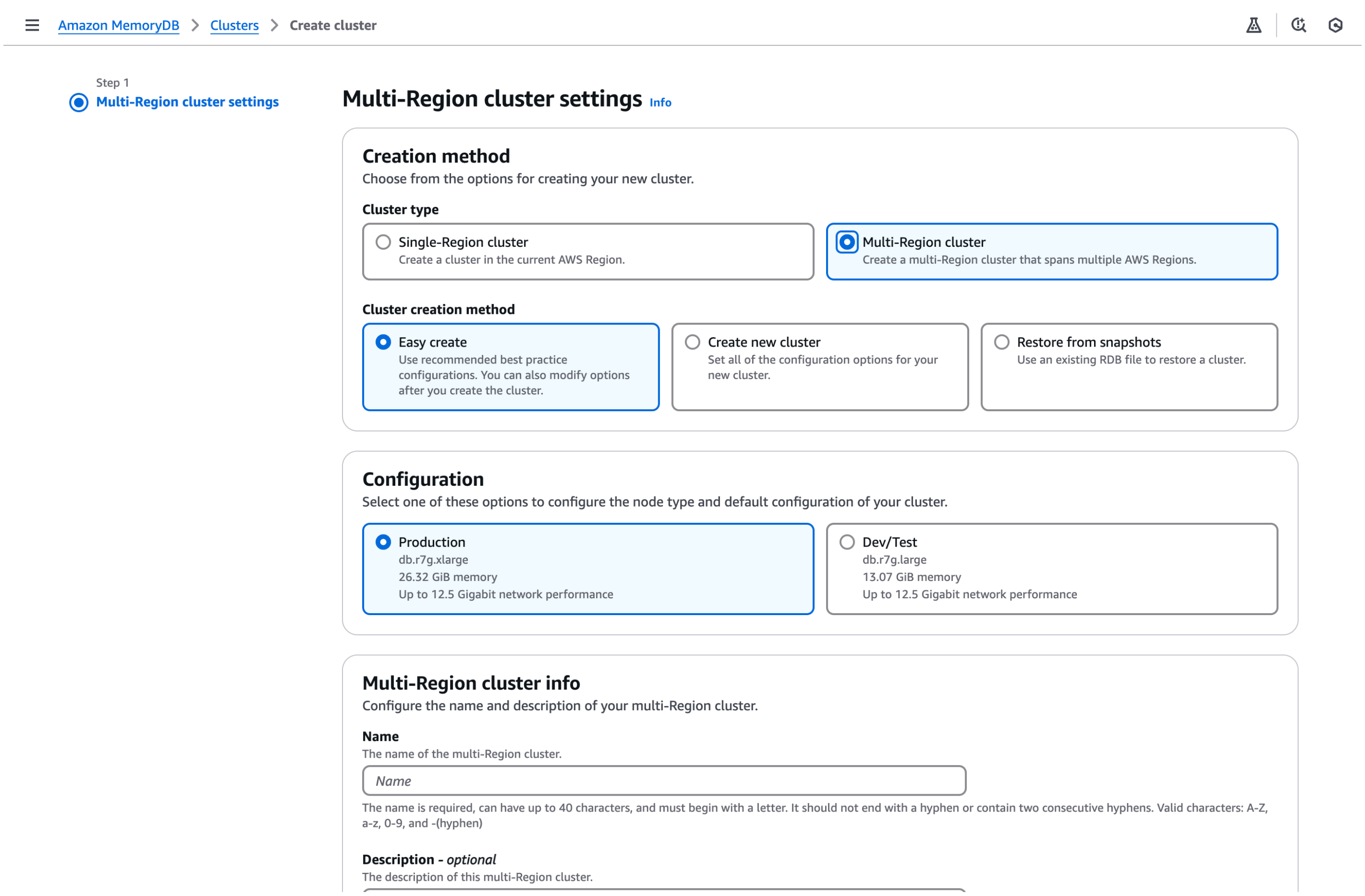
콘솔에서 MemoryDB Multi-Region 사용

다음은 콘솔에서 MemoryDB Multi-Region을 사용하는 방법입니다.

주제

MemoryDB Multi-Region에서 새 클러스터 생성

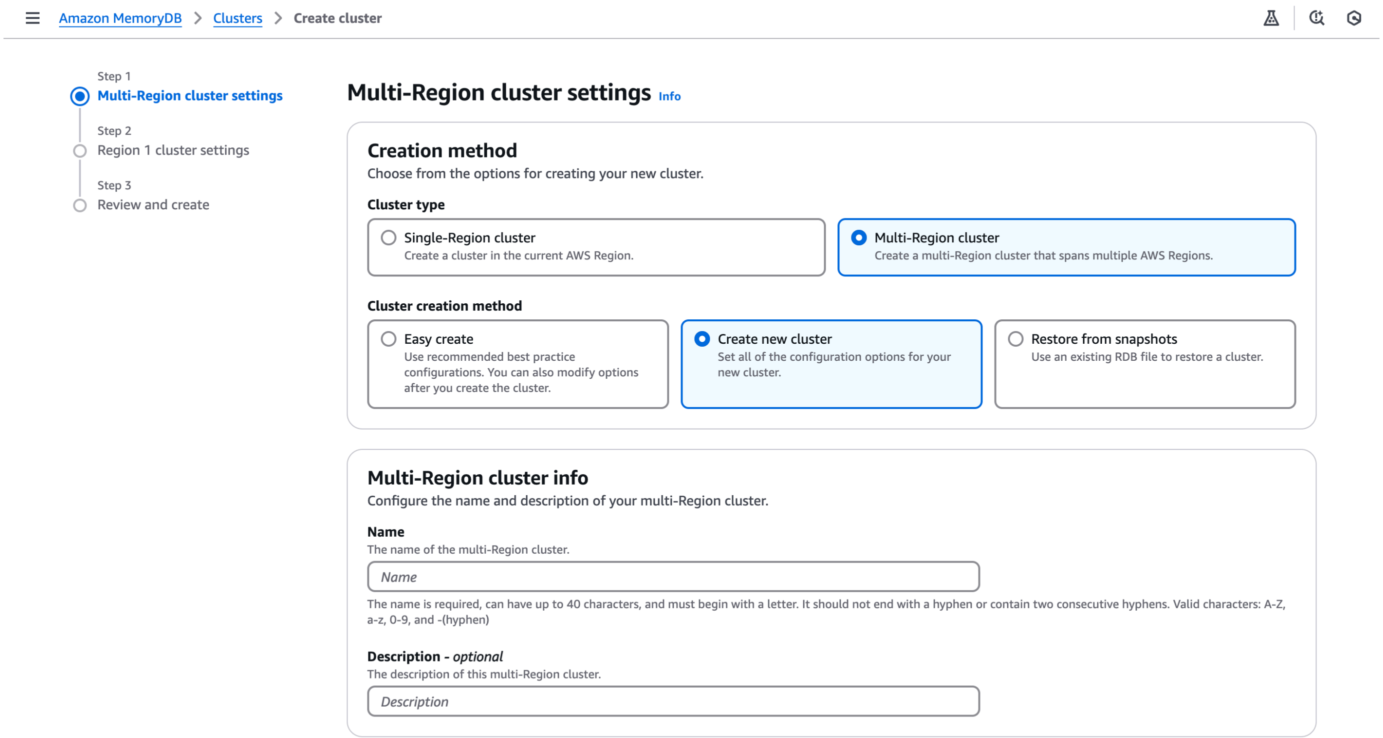
1. 클러스터 목록 또는 대시보드에서 클러스터 생성 섹션으로 이동합니다.



2. 클러스터 유형 필드에서 Multi-Region 클러스터를 선택합니다.
3. 클러스터 생성 방법 필드에서 간편 생성을 선택합니다.
4. 이름 및 설명을 입력하고 기본값을 확인한 다음 생성을 선택합니다.

클러스터 생성 및 구성

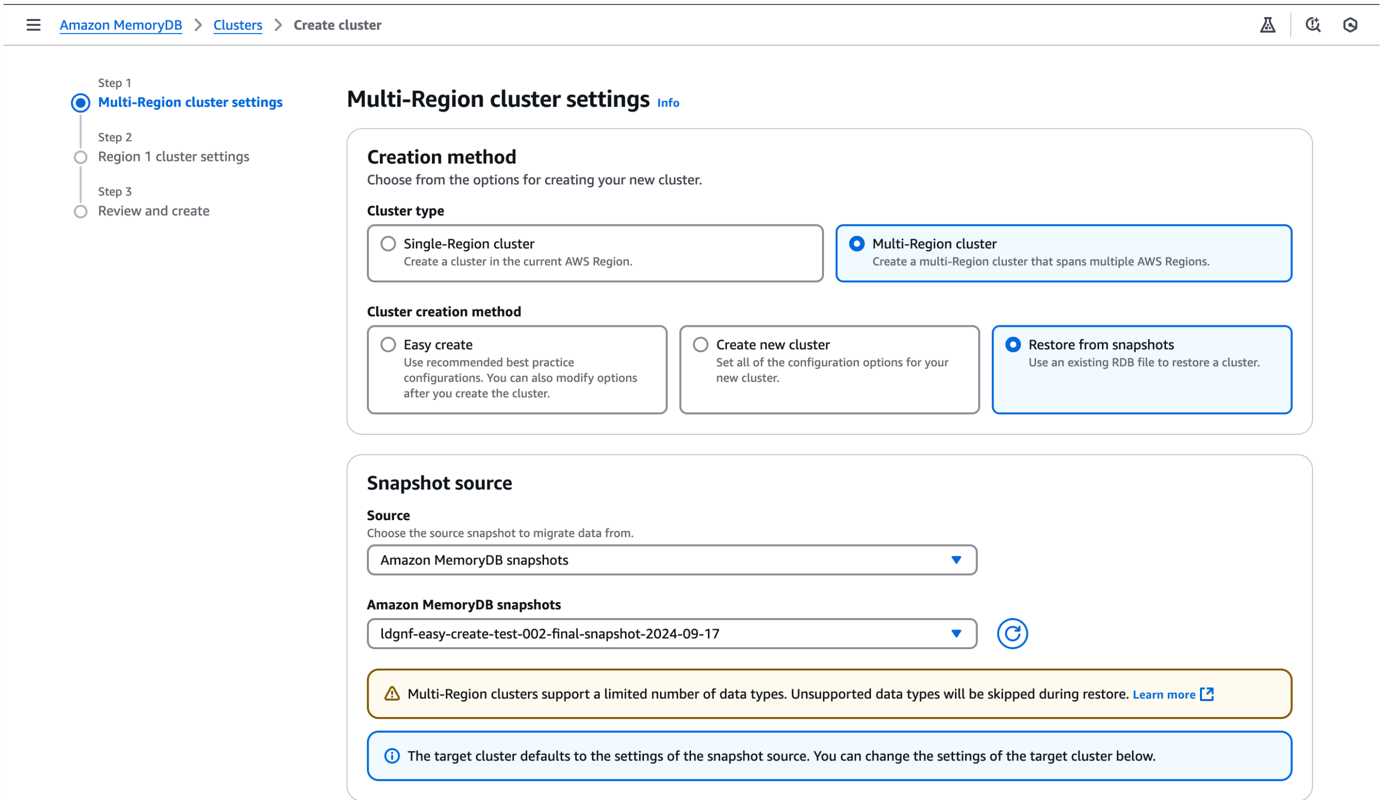
1. 클러스터 목록 또는 대시보드에서 클러스터 생성 섹션으로 이동합니다.



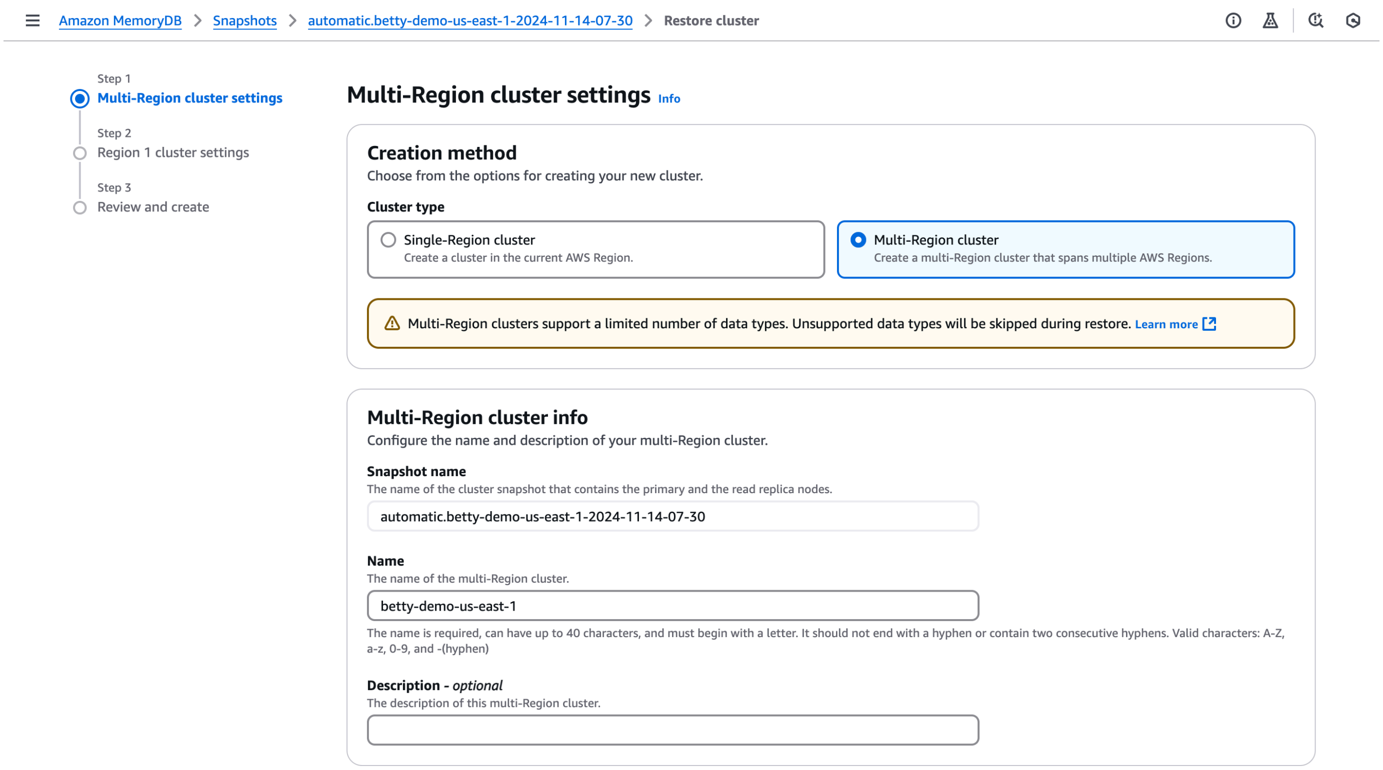
2. 클러스터 유형 필드에서 Multi-Region 클러스터를 선택합니다.
3. 클러스터 생성 방법 필드에서 새 클러스터 생성을 선택합니다.
4. 이름 및 설명을 입력하고 값을 확인한 다음 생성을 선택합니다.

Multi-Region 클러스터 내의 새 클러스터 또는 기존 클러스터에 스냅샷 복원

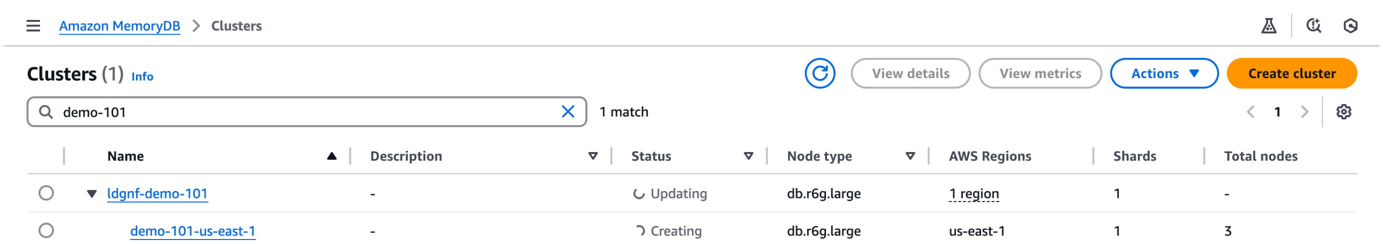
1. 클러스터 목록 또는 대시보드에서 클러스터 생성 섹션으로 이동합니다.



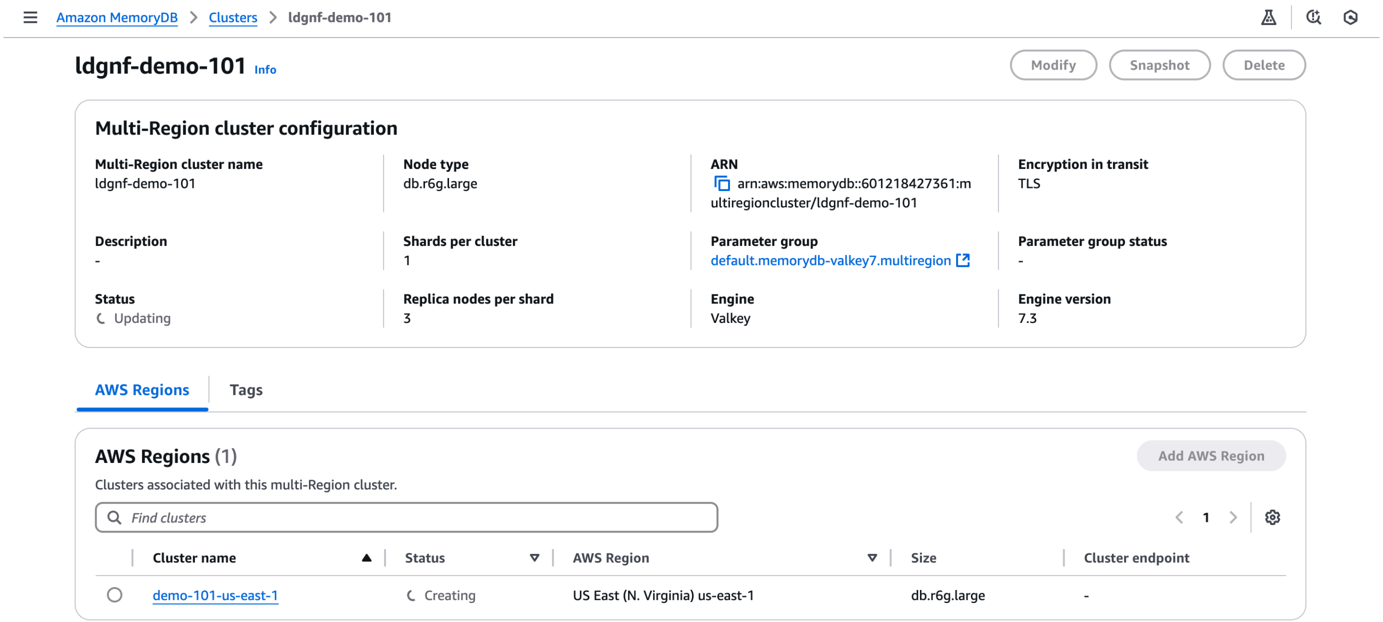
2. 클러스터 유형 필드에서 Multi-Region 클러스터를 선택합니다.
3. 클러스터 생성 방법 필드에서 스냅샷에서 복원을 선택합니다.
4. 소스 스냅샷을 선택한 다음 필수 필드를 입력합니다. 선택 사항을 검토한 다음 복원을 선택합니다.



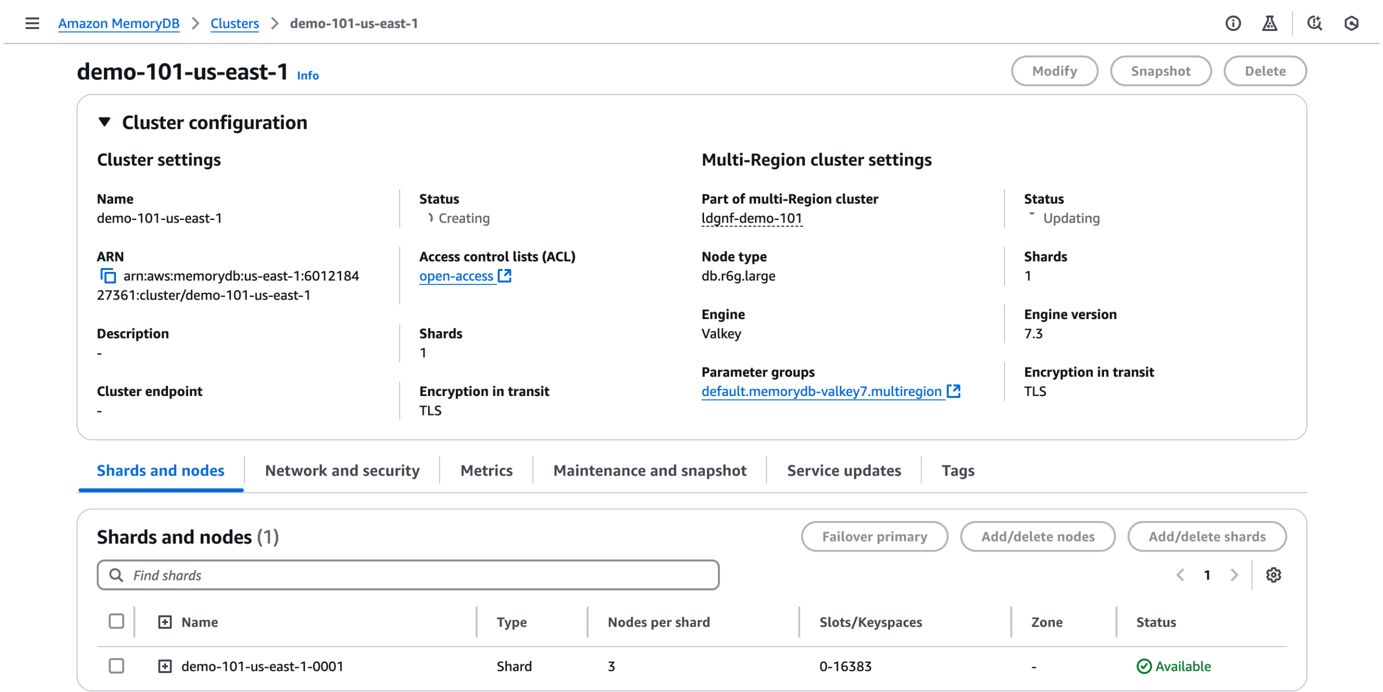
5. Multi-Region 클러스터를 보려면 클러스터 섹션으로 이동합니다.



6. 이제 대상 다중 리전 클러스터 이름을 선택합니다.

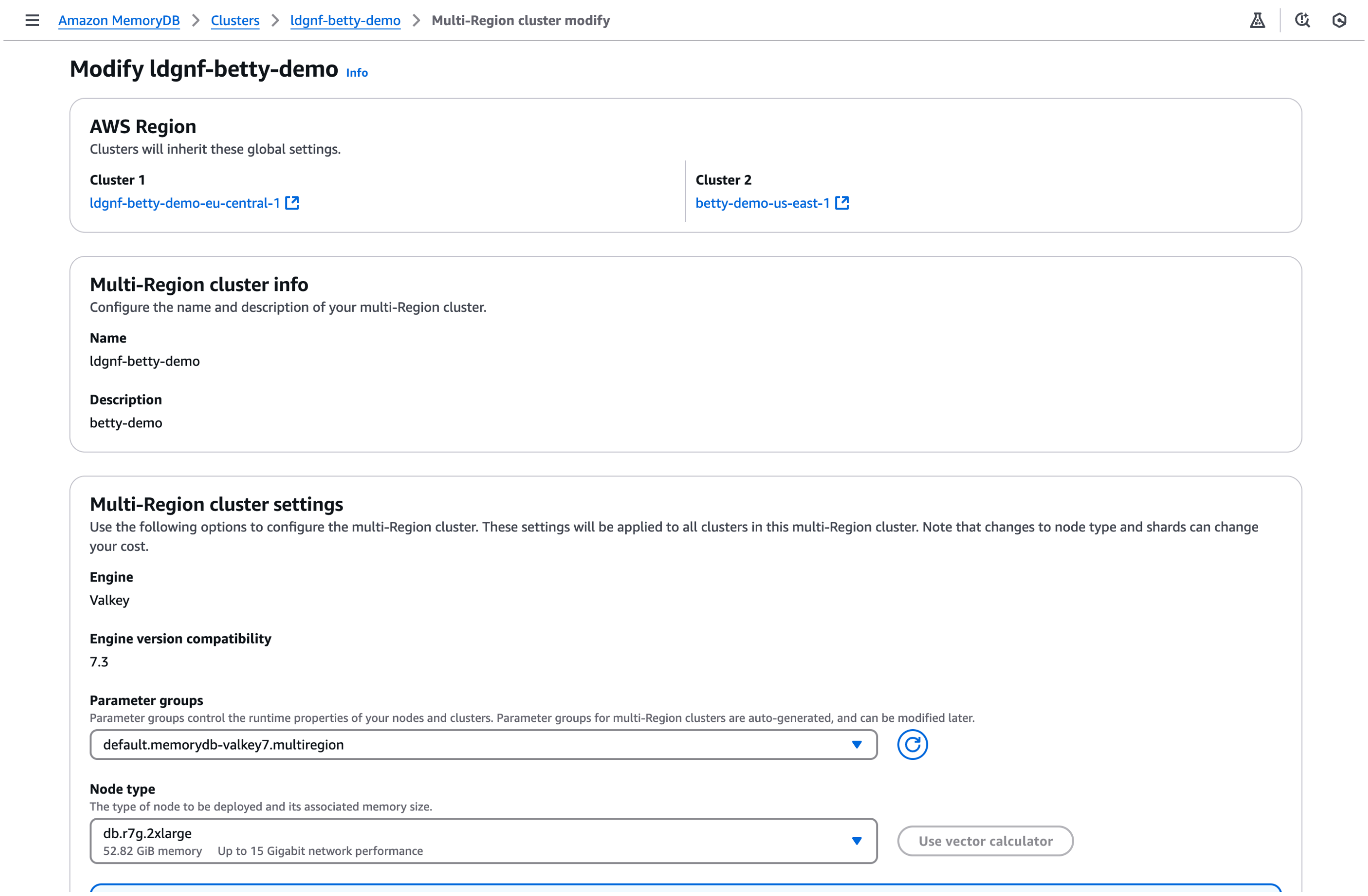


7. 이제 대상 리전 클러스터 이름을 선택합니다.



MemoryDB Multi-Region에서 클러스터 수정

1. 데이터 섹션으로 이동합니다. 현재 클러스터가 모두 표시되어야 합니다.

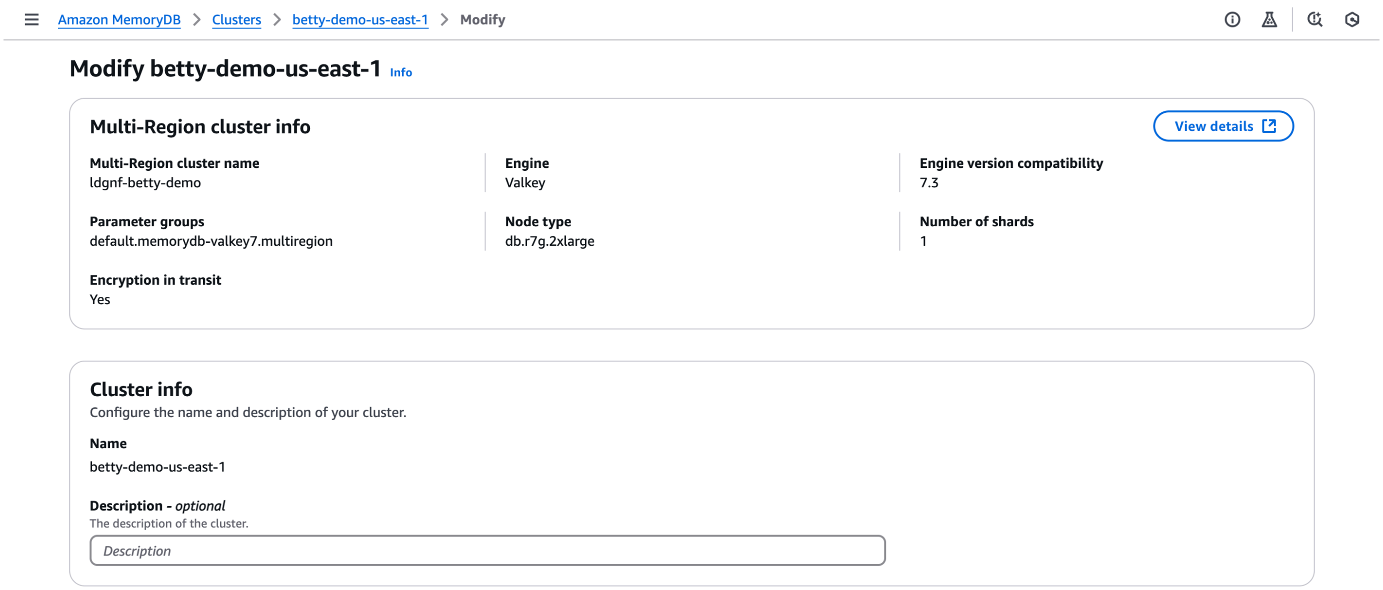


그런 다음 수정하려는 클러스터 유형에 따라 다음 단계 중에서 선택합니다.

2. Muti-Region 클러스터로 단일 클러스터를 수정하려면 먼저 해당 클러스터가 속한 다중 리전을 선택합니다. 그런 다음 작업(오른쪽 상단)에서 편집 버튼을 선택합니다. 그런 다음 대상 단일 클러스터를 선택합니다. 세부 정보 페이지에서 이 클러스터를 수정할 수도 있습니다.

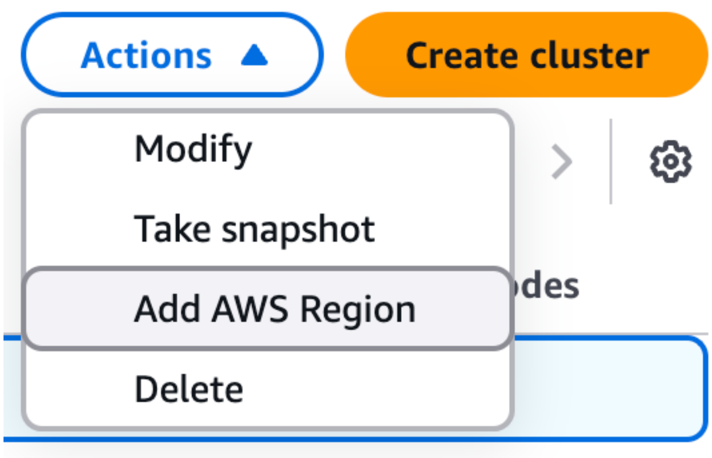
리전 클러스터 수정

1. 다중 리전 클러스터를 수정하려면 대상 다중 리전 클러스터 이름을 선택합니다.

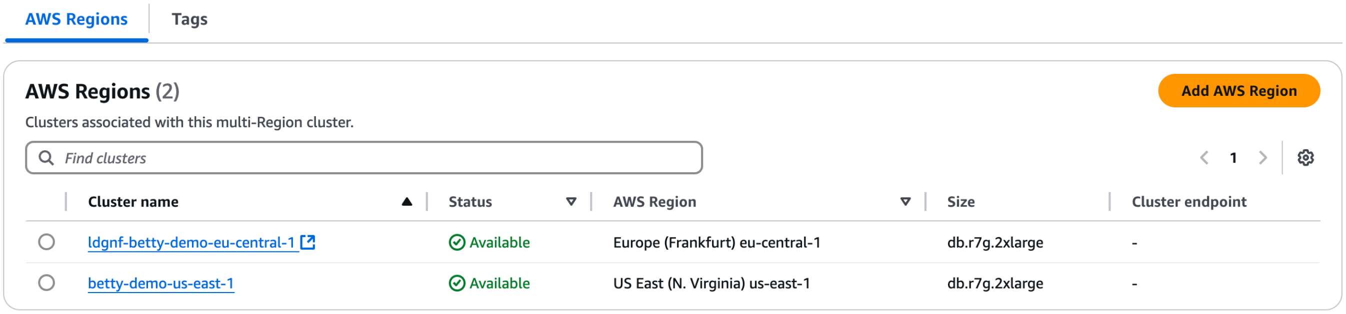


그런 다음 클러스터를 선택하고 작업(오른쪽 상단) 또는 세부 정보 페이지에서 편집 버튼을 선택합니다.

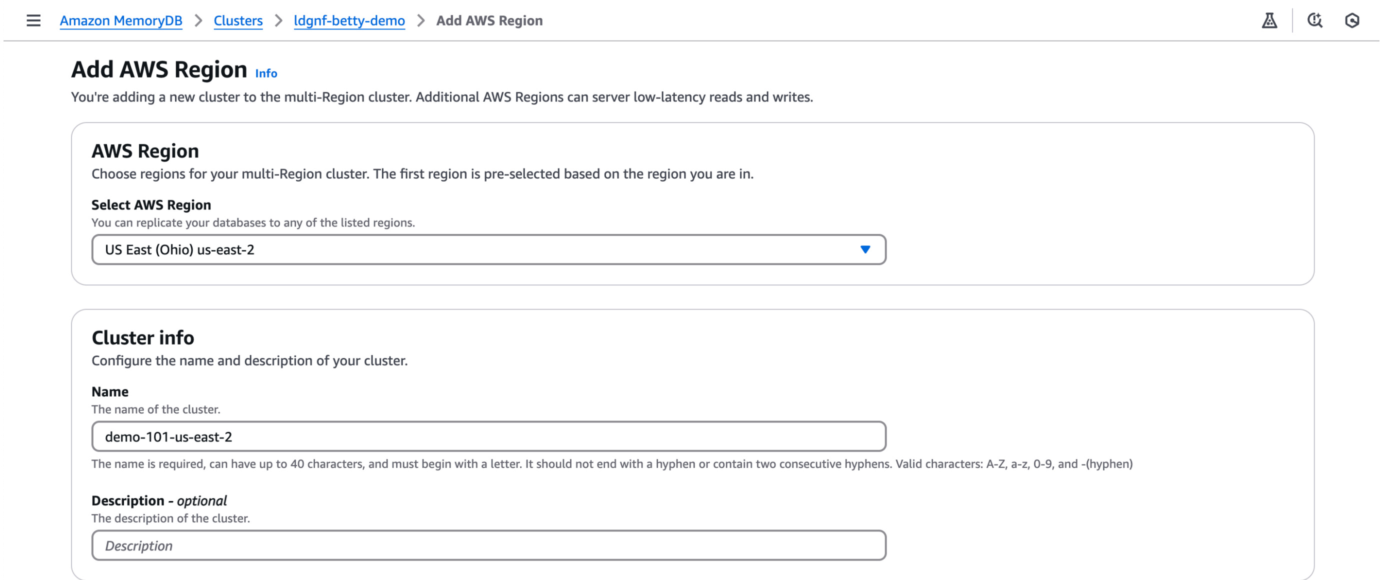
2. 리전 클러스터를 추가하려면 선택한 대상 다중 리전 클러스터를 선택한 다음 작업 드롭다운으로 이동하여 AWS 리전 추가를 선택합니다. AWS 리전의 세부 정보 페이지로 이동하여 대상 다중 리전 클러스터를 선택하고 거기서 추가할 수도 있습니다.



3. 리전을 추가하려면 대상 리전을 선택합니다. 그런 다음 필수 정보를 입력하고 AWS 리전 추가를 선택합니다.

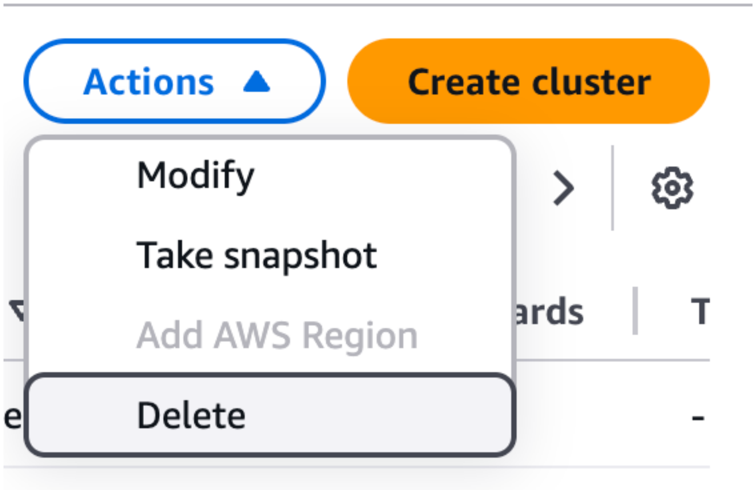


4. 빈 다중 리전 클러스터에 새 리전 클러스터를 추가하려면 다중 리전 클러스터 생성과 동일한 옵션이 표시됩니다. 유일한 차이점은 다중 리전 클러스터 정보가 이미 존재한다는 것입니다.

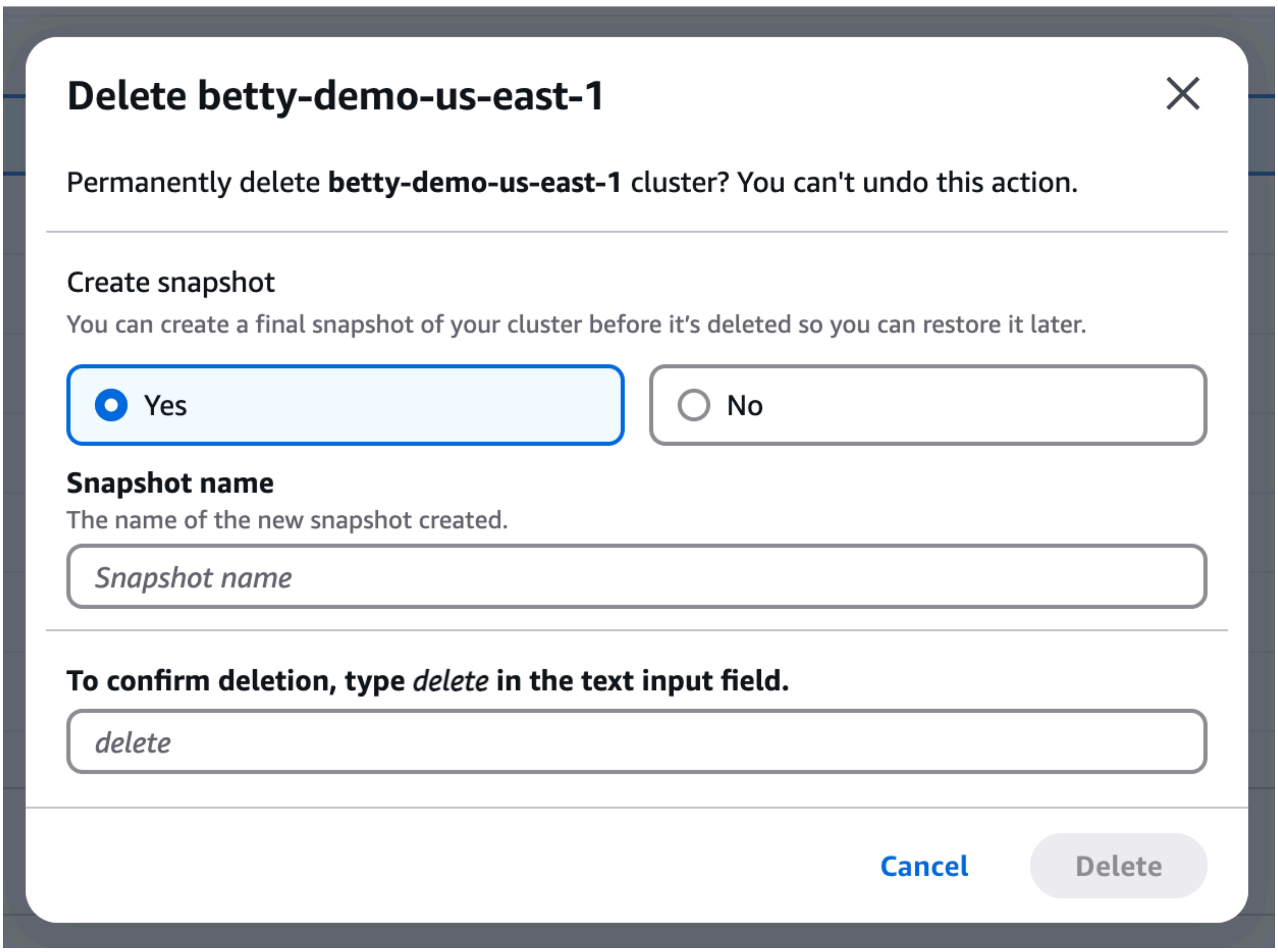


MemoryDB Multi-Region에서 클러스터 삭제

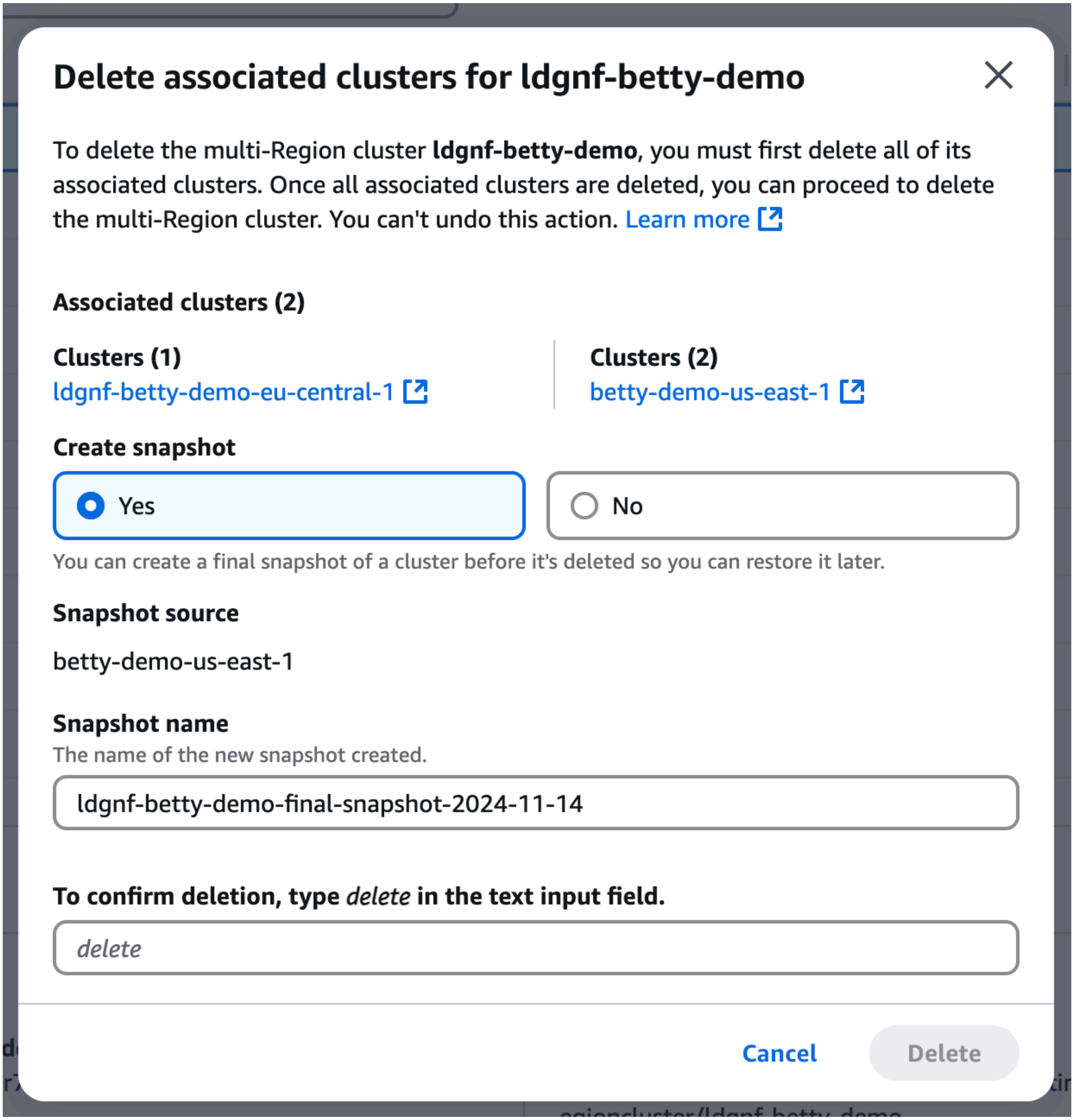
1. 리전에서 단일 클러스터를 삭제하려면 대상 리전 클러스터를 선택합니다. 그런 다음 작업 메뉴 드롭다운으로 이동하여 개별 클러스터를 선택하고 삭제를 선택합니다.



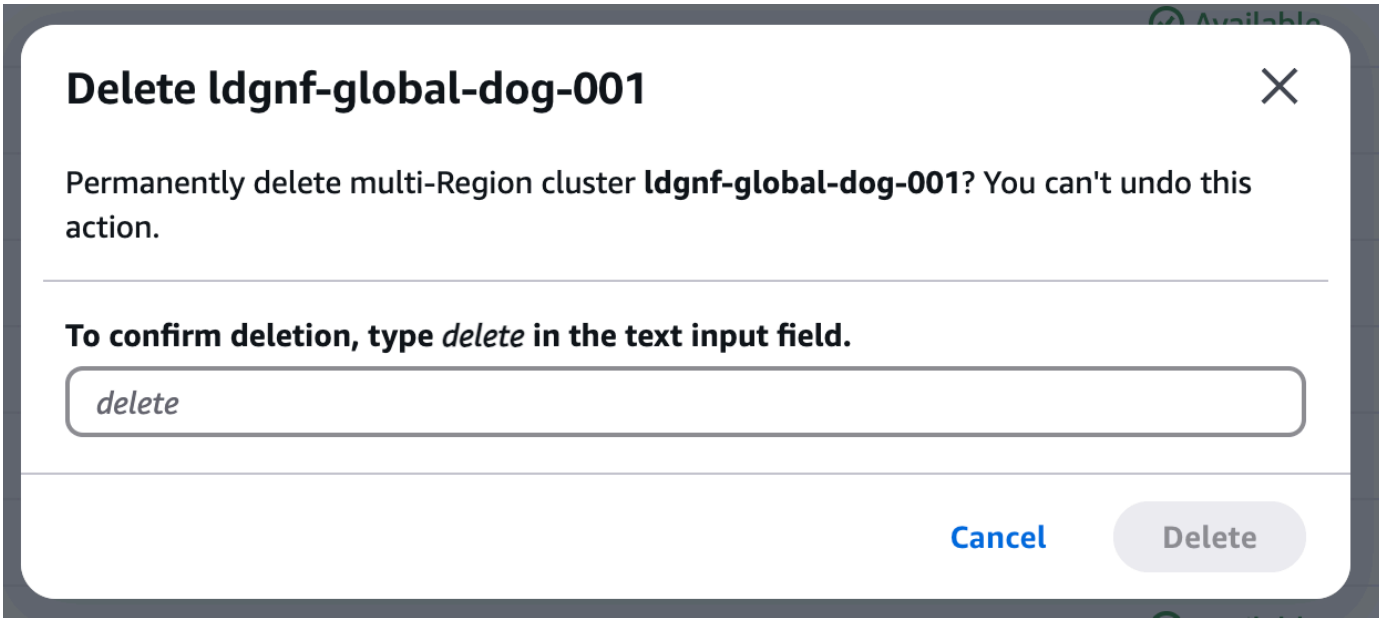
삭제하기 전에 스냅샷을 생성하는 옵션을 포함한 확인 창이 표시됩니다. 그래도 삭제하려면 텍스트 필드에 "삭제"를 입력한 다음 삭제를 선택합니다.



2. 다중 리전 클러스터와 연결된 모든 리전 클러스터를 삭제하려면 하나 이상의 클러스터가 있는 대상 다중 리전 클러스터를 선택합니다. 그런 다음 대상 다중 리전 클러스터를 선택한 상태에서 작업 메뉴 드롭다운으로 이동하여 삭제를 선택합니다.



3. 전체 다중 리전 클러스터를 삭제하려면 대상 빈 다중 리전 클러스터를 선택합니다. 그런 다음 작업 메뉴 드롭다운으로 이동하여 삭제를 선택합니다.



CLI에서 MemoryDB Multi-Region 사용

다음은 CLI에서 MemoryDB Multi-Region을 사용하는 방법입니다.

> **Note**
> MemoryDB Multi-Region은 노드 유형 db.r7g.xlarge 이상만 지원합니다.

MemoryDBMulti 리전을 사용하여 클러스터 생성

다중 리전 클러스터 생성

```
aws memorydb create-multi-region-cluster \
 --multi-region-cluster-name-suffix my-multi-region-cluster \
 --node-type db.r7g.xlarge \
 --engine valkey \
 --region us-east-1
```

미국 동부(버지니아 북부) 리전에서 리전 클러스터 생성

```
aws memorydb create-cluster \
 --cluster-name my-cluster \
 --multi-region-cluster-name my-multi-region-cluster \
```

```
 --node-type db.r7g.xlarge \
 --acl-name open-access \
 --region us-east-1 \
```

유럽(아일랜드) 리전에서 리전 클러스터 생성

```
aws memorydb create-cluster \
 --cluster-name my-cluster \
 --multi-region-cluster-name my-multi-region-cluster \
 --node-type db.r7g.xlarge \
 --acl-name open-access \
 --region eu-west-1 \
```

모든 리전의 다중 리전 클러스터 설명

```
aws memorydb describe-multi-region-cluster \
 --multi-region-cluster-name my-multi-region-cluster \
 --region eu-west-1
```

다중 리전 클러스터 업데이트

노드 유형 수정

```
aws memorydb update-multi-region-cluster \
 --multi-region-cluster-name my-multi-region-cluster \
 --node-type db.r7g.4xlarge \
 --region us-east-1
```

샤드 수 수정

```
aws memorydb update-multi-region-cluster \
 --multi-region-cluster-name my-multi-region-cluster \
 --shard-configuration \
 ShardCount=3 \
 --update-strategy COORDINATED \
 --region us-east-1
```

MemoryDB 클러스터 규모 조정

먼저 `list-allowed-node-type-updates` 명령을 사용하여 스케일 업하거나 스케일 다운할 수 있는 노드를 나열합니다.

```
aws memorydb list-allowed-node-type-updates \
 --cluster-name my-cluster-name
```

이렇게 하면 확장 또는 축소할 수 있는 노드 목록이 제공됩니다. 다음과 같이 `update-cluster` 명령을 사용하여 사물을 업데이트할 수 있습니다.

```
aws memorydb update-cluster  \
 --cluster-name my-cluster \
 --node-type db.r6g.2xlarge
```

다중 리전 KMS 키에 대한 자세한 내용은 [MemoryDB Multi-Region을 사용한 규모 조정](#) 단원을 참조하세요.

MemoryDB Multi-Region에서 클러스터 삭제

리전 클러스터 삭제

```
aws memorydb delete-cluster \
 --cluster-name my-cluster \
 --multi-region-cluster-name my-multi-region-cluster \
 --region us-east-1
```

다중 리전 클러스터 삭제

```
aws memorydb delete-multi-region-cluster \
 --multi-region-cluster-name my-multi-region-cluster \
 --region us-east-1
```

MemoryDB Multi-Region 모니터링

Amazon CloudWatch를 사용하여 Multi-Region 클러스터의 동작과 성능을 모니터링할 수 있습니다. MemoryDB는 Multi-Region 클러스터 내의 각 리전 클러스터에 대한 `MultiRegionClusterReplicationLag` 지표를 게시합니다.

`MultiRegionClusterReplicationLag`는 업데이트가 원격 Multi-Region 리전 클러스터 다중 AZ 트랜잭션 로그에 기록되는 시점과 해당 업데이트가 로컬 Multi-Region 리전 클러스터의 프라이머리 노드에 기록되는 시점 사이의 경과 시간을 보여줍니다. 이 지표는 밀리초 단위로 표시되며 샤드 수준에서 모든 소스 및 대상 리전 페어에 대해 내보내집니다.

정상 작동 중에는 `MultiRegionClusterReplicationLag`가 상당히 일정해야 합니다. `MultiRegionClusterReplicationLag` 값이 상승하면 한 리전 클러스터의 업데이트 내용이 다른 리전 클러스터로 시기 적절하게 전파되지 않는다는 것을 나타낼 수 있습니다. 시간이 지날수록 다른 리전 클러스터가 더 이상 지속적으로 업데이트 내용을 받지 않기 때문에 뒤처질 수 있습니다.

`MultiRegionClusterReplicationLag`는 AWS 리전이 격리되거나 성능이 저하되고 해당 리전에 리전 클러스터가 있는 경우에 증가할 수 있습니다. 이 경우 애플리케이션의 읽기 및 쓰기 작업을 다른 정상 AWS 리전으로 일시적으로 리디렉션할 수 있습니다.

MemoryDB Multi-Region을 사용한 규모 조정

클러스터에 대한 수요 변화에 따라 MemoryDB 클러스터 내 샤드 수 또는 노드 유형을 변경해 성능을 향상시키거나 비용을 줄이도록 결정할 수 있습니다. MemoryDB Multi-Region 클러스터를 확장하면 해당 클러스터의 모든 리전 클러스터가 확장됩니다. MemoryDB Multi-Region 클러스터는 온라인 리샤딩을 지원합니다. MemoryDB Multi-Region 클러스터는 오프라인 리샤딩을 지원하지 않습니다.

클러스터를 다시 조정하도록 결정할 수 있는 조건은 다음과 같습니다.

- 메모리 부족:

  리전 클러스터의 노드에서 메모리가 부족하면 데이터를 저장 및 요청 처리에 더 많은 리소스를 사용하도록 스케일 아웃 또는 스케일 업을 결정할 수 있습니다.

  다음과 같은 지표를 모니터링해 노드에서 메모리가 부족한지 확인할 수 있습니다. FreeableMemory, SwapUsage, BytesUsedForMemoryDB, MultiRegionClusterReplicationLag
- CPU 또는 네트워크 병목 현상:

  클러스터에서 지연 시간/처리량 문제가 발생하면 문제를 해결하기 위해 스케일 아웃 또는 스케일 업이 필요할 수 있습니다.

  다음과 같은 지표를 모니터링하여 지연 시간 및 처리량 수준을 모니터링할 수 있습니다. `CPUUtilization`, `NetworkBytesIn`, `NetworkBytesOut`, `CurrConnections`, `NewConnections`, `and MultiRegionClusterReplicationLag`.
- 클러스터가 과도하게 조정됨:

  스케일 인 또는 스케일 다운과 같은 클러스터에 대한 현재 수요는 성능을 저하시키지 않고 비용을 줄입니다.

다음과 같은 지표를 사용하여 클러스터의 사용을 모니터링해 안전하게 스케일 인 또는 스케일 다운할 수 있는지 확인할 수 있습니다. FreeableMemory, SwapUsage, BytesUsedForMemoryDB, CPUUtilization, NetworkBytesIn, NetworkBytesOut, CurrConnections, NewConnections, MultiRegionClusterReplicationLag

MemoryDB 클러스터를 조정하는 방법에는 수평적 스케일링과 수직적 스케일링이라는 두 가지 방법이 있습니다.

- 수평적 스케일링에서는 샤드를 추가 또는 제거하여 MemoryDB Multi-Region 클러스터 내 샤드 수를 변경할 수 있습니다. 온라인 리샤딩 프로세스를 통해 리전 클러스터가 들어오는 요청을 계속 처리하는 동안 확장/축소할 수 있습니다.
- 수직적 스케일링에서는 노드 유형을 변경하여 MemoryDB Multi-Region 클러스터의 크기를 조정합니다. 온라인 수직적 스케일링을 통해 리전 클러스터가 들어오는 요청을 계속 처리하는 동안 확장/축소할 수 있습니다.

조정은 기본적으로 "협정된" 업데이트 전략을 사용합니다. 즉, 모든 리전 클러스터가 성공적으로 스케일링되지 않는다면 리전 클러스터가 스케일링되지 않습니다.

스케일 아웃 작업은 "협정되지 않은" 업데이트 전략도 지원합니다. 즉, 일부 리전 클러스터는 성공적으로 스케일 아웃될 수 있지만 일부 리전 클러스터는 스케일 아웃 시도에 실패합니다. 한 리전 클러스터 스케일 아웃이 성공하면 다른 모든 리전 클러스터는 다른 각 스케일 아웃도 성공할 때까지 스케일 아웃을 계속 재시도합니다.

모든 리전 클러스터가 스케일 아웃에 실패하면 다중 리전 클러스터는 "협정되지 않은" 스케일 아웃에 실패합니다.

> **Note**
>
> "협정되지 않은" 스케일 아웃은 리전 클러스터가 서로 다른 시간에 스케일 아웃될 때 리전 클러스터 간에 용량 불균형 상태를 연장할 수 있습니다. 이로 인해 MultiRegionClusterReplicationLag 지표가 증가하고 리전 클러스터 데이터가 장기간 발산될 수 있습니다.

MemoryDB Multi-Region 클러스터 리전 클러스터는 복제본 노드 수에 대해 서로 다른 구성을 가질 수 있지만 리전 클러스터의 모든 샤드에는 동일한 수의 복제본 노드가 있습니다.

MemoryDB Multi-Region 클러스터의 크기와 메모리 용량을 축소하려면 스케일 인 또는 스케일 다운을 통해 새 구성에 데이터를 위한 충분한 메모리와 여유 IPs, 충분한 엔진 오버헤드가 있는지, 리전 클러스터에 대한 MultiRegionClusterReplicationLag 지표가 초 또는 분 범위 내에 있는지 확인합니다.

AWS Management 콘솔, AWS CLI, MemoryDB API를 사용하여 MemoryDB Multi-Region 클러스터를 수평적 스케일링 및 수직적 스케일링할 수 있습니다.

지원되는 명령과 지원되지 않는 명령

지원되는 명령

> **Note**
>
> - SET 명령은 현재 EX, PX, EXAT, PXAT, KEEPTTL 옵션을 지원하지 않습니다.
> - RESTORE 명령은 TTL을 0이 아닌 값으로 설정하는 것을 지원하지 않습니다. ABSTTL, IDLETIME 및 FREQ 옵션도 지원되지 않습니다.

데이터 유형	명령
String	SET*, DECR, DECRBY, GET, GETRANGE, SUBSTR, GETDEL, GETSET, INCR, INCRBY, INCRBYFLOAT, MGET, MSET, MSETNX, SETNX, STRLEN, LCS
Hash	HINCRBY, HINCRBYFLOAT, HDEL, HSET, HMSET, HGET, HEXISTS, HLEN, HKEYS, HVALS, HGETALL, HMGET, HSTRLEN, HSETNX, HRANDFIELD, HSCAN
Set	SADD, SREM, SISMEMBER, SMISMEMBE R, SCARD, SMEMBERS, SRANDMEMBER, SSCAN, SUNION, SINTERCARD, SINTER, SDIFF, SPOP

데이터 유형	명령
Sorted Set	ZADD, ZINCRBY, ZSCORE, ZMSCORE, ZCARD, ZRANK, ZREVRANK, ZRANGE, ZRANGEBYSCORE, ZRANGEBYLEX, ZREVRANGE, ZREVRANGEBYLEX, ZREVRANGEBYSCORE, ZREMRANGE BYLEX, ZREMRANGEBYSCORE, ZREMRANGEBYRANK, ZUNION, ZINTER, ZINTERCARD, ZDIFF, ZLEXCOUNT, ZCOUNT, ZREM, ZMPOP, ZPOPMIN, ZPOPMAX, ZSCAN, ZRANDMEMBER
Generic	SCAN, DEL, UNLINK, DUMP, RESTORE**, EXISTS, KEYS, RANDOMKEY, TYPE

지원되지 않는 명령

지원되지 않는 명령의 일반적인 범주는 지원되지 않는 데이터 유형(Bitmaps, Hyperloglog, List, Geospatial, Stream), TTL 관련 명령, 차단 명령 및 함수 관련 명령입니다. 전체 목록은 다음과 같습니다.

데이터 유형	명령
String	APPEND, GETEX, SETEX, SETRANGE
Bitmap	BITCOUNT, BITFIELD, BITFIELD_RO, BITOP, BITPOS, GETBIT, SETBIT
Hyperloglog	PFADD, PFCOUNT, PFDEBUG, PFMERGE, PFSELFTEST
List	BLMOVE, BLMPOP, BLPOP, BRPOP, BRPOPLPUSH, LINDEX, LINSERT, LLEN, LMOVE, LMPOP, LPOP, LPOS, PUSH, LPUSHX, LRANGE, LREM, LSET, LTRIM, RPOP, RPOPLPUSH, RPUSH, RPUSHX

데이터 유형	명령
Set	SMOVE, SUNIONSTORE, SDIFFSTORE, SINTERSTORE
Sorted Set	BZMPOP, BZPOPMAX, BZPOPMIN, ZDIFFSTORE, ZINTERSTORE, ZRANGESTORE, ZUNIONSTORE
Geospatial	GEOADD, GEODIST, GEOHASH, GEOPOS, GEORADIUS, GEORADIUS_RO, GEORADIUSBYMEMBER, GEORADIUSBYMEMBER_RO, GEOSEARCH, GEOSEARCHSTORE
Stream	XACK, XADD, XAUTOCLAIM, XCLAIM, XDEL, XLEN, XPENDING, XRANGE, XREAD, XREADGROUP, XREVRANGE, XSETID, XTRIM, XGROUP, XINFO
Generic	COPY, FLUSHDB, FLUSHALL, MOVE, RENAME, RENAMENX, SORT, SORT_RO, SWAPDB, OBJECT, FUNCTION, FCALL, FCALL_RO, EXPIRE, EXPIREAT, EXPIRETIME, PERSIST, PEXPIRE, PEXPIREAT, PEXPIRETIME, PSETEX, PTTL, TTL

MemoryDB의 보안

의 클라우드 보안 AWS 이 최우선 순위입니다. AWS 고객은 보안에 가장 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 이점을 누릴 수 있습니다.

보안은 AWS 와 사용자 간의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS 는 AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다. AWS 또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 타사 감사자는 AWS 규정 준수 프로그램 일환으로 보안의 효과를 정기적으로 테스트하고 확인합니다. MemoryDB에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램 AWS 제공 범위 내 서비스규정 준수 프로그램 .
- 클라우드의 보안 - 사용자의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 여러분은 데이터의 민감도, 회사 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 MemoryDB 사용 시 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 보안 및 규정 준수 목표에 맞게 MemoryDB를 구성하는 방법을 보여줍니다. 또한 MemoryDB 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법을 알아봅니다.

내용

MemoryDB의 데이터 보호

AWS공동 책임 모델의 데이터 보호에 적용됩니다. 이 모델에 설명된 대로 AWS는 모든를 실행하는 글로벌 인프라를 보호할 책임이 있습니다AWS 클라우드. 사용자는 이 인프라에 호스팅되는 콘텐츠에 대한 통제 권한을 유지할 책임이 있습니다. 사용하는 AWS 서비스의 보안 구성과 관리 태스크에 대한 책

임도 사용자에게 있습니다. 데이터 프라이버시에 관한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터 보호를 위해 자격 증명을 보호하고AWS 계정 AWS IAM Identity Center또는 AWS Identity and Access Management(IAM)를 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용합니다.
- SSL/TLS를 사용하여 AWS리소스와 통신합니다. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다AWS CloudTrail. CloudTrail 추적을 사용하여 AWS활동을 캡처하는 방법에 대한 자세한 내용은 AWS CloudTrail사용 설명서의 CloudTrail 추적 작업을 참조하세요.
- 내의 모든 기본 보안 제어와 함께 AWS암호화 솔루션을 사용합니다AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용합니다.
- 명령줄 인터페이스 또는 API를 AWS통해에 액세스할 때 FIPS 140-3 검증 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 연방 정보 처리 표준(FIPS) 140-3을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 또는 기타AWS 서비스에서 콘솔, API AWS CLI또는 AWSSDKs를 사용하여 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명을 URL에 포함해서는 안 됩니다.

MemoryDB의 데이터 보안

데이터를 안전하게 보관하기 위해 MemoryDB 및 Amazon EC2에서는 서버의 데이터에 대한 무단 액세스를 방지하는 메커니즘을 제공합니다.

MemoryDB는 클러스터의 데이터에 대한 암호화 기능도 제공합니다.

- 전송 중 데이터 암호화는 클러스터 내 노드 사이 또는 클러스터와 애플리케이션 사이와 같이 한 위치에서 다른 위치로 데이터가 이동 중인 경우 항상 데이터를 암호화합니다.

- 미사용 데이터 암호화는 스냅샷 작업 중 트랜잭션 로그 및 디스크 내 데이터를 암호화합니다.

액세스 제어 목록(ACL)을 사용하여 사용자 인증을(를) 사용하여 클러스터에 대한 사용자 액세스를 제어할 수도 있습니다.

주제

- MemoryDB의 미사용 데이터 암호화
- MemoryDB에서 전송 중 데이터 암호화(TLS)
- 액세스 제어 목록(ACL)을 사용하여 사용자 인증
- IAM을 통한 인증

MemoryDB의 미사용 데이터 암호화

데이터를 안전하게 보관하기 위해 MemoryDB 및 Amazon S3에서는 클러스터에 대한 액세스를 제한하는 다양한 방식을 제공합니다. 자세한 내용은 MemoryDB 및 Amazon VPC 및 MemoryDB의 ID 및 액세스 관리(을)를 참조하세요.

영구 데이터를 암호화하여 데이터 보안을 강화하기 위해 MemoryDB 미사용 데이터 암호화는 항상 활성화됩니다. 다음과 같은 측면을 암호화합니다.

- 트랜잭션 로그의 데이터
- 동기화, 스냅샷 및 스왑 작업 중 디스크
- Amazon S3에 저장된 스냅샷

MemoryDB는 기본(서비스 관리형) 유휴 데이터 암호화와 더불어 AWS Key Management Service(KMS)에서 자체 대칭 고객 관리형 고객 마스터 키를 사용할 수 있는 기능을 제공합니다.

데이터 계층화가 활성화된 클러스터의 SSD(Solid-State Drive)에 저장된 데이터는 기본적으로 항상 암호화됩니다.

전송 중 데이터 암호화와 관련된 자세한 내용은 MemoryDB에서 전송 중 데이터 암호화(TLS)을 참조하세요.

주제

- AWS KMS에서 고객 관리형 키 사용
- 참고

AWS KMS에서 고객 관리형 키 사용

MemoryDB는 저장된 데이터 암호화에 대한 대칭 고객 관리형 루트(KMS 키)를 지원합니다. 고객 관리형 KMS 키는 사용자가 생성, 소유 및 관리하는 AWS 계정의 암호화 키입니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 고객 루트 키를 참조하세요. MemoryDB와 함께 사용하기 전에 AWS KMS에서 키를 생성해야 합니다.

AWS KMS 루트 키 생성 방법을 알아보려면 AWS Key Management Service 개발자 안내서에서 키 생성을 참조하세요.

MemoryDB를 사용하면 AWS KMS와 통합할 수 있습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 권한 부여 사용을 참조하세요. MemoryDB와 AWS KMS의 통합을 활성화하기 위해 고객이 별도로 취해야 할 조치는 없습니다.

`kms:ViaService` 조건 키는 AWS KMS 키의 사용을 지정된 AWS 서비스로부터의 요청으로 제한합니다. MemoryDB에서 `kms:ViaService`를 사용하려면 조건 키 값 `memorydb.amazon_region.amazonaws.com`에 ViaService 이름을 모두 포함시켜야 합니다. 자세한 내용은 kms:ViaService 섹션을 참조하세요.

AWS CloudTrail을 사용하여 MemoryDB가 사용자 대신 AWS Key Management Service에 전송하는 요청을 추적할 수 있습니다. 고객 관리형 키와 관련된 AWS Key Management Service에 대한 모든 API 호출에는 해당 CloudTrail 로그가 있습니다. ListGlants KMS API 직접 호출을 통해 MemoryDB가 생성하는 그랜트를 볼 수도 있습니다.

고객 관리형 키를 사용하여 클러스터를 암호화하면 클러스터의 모든 스냅샷이 다음과 같이 암호화됩니다.

- 자동 일일 스냅샷은 클러스터와 연결된 고객 관리형 키를 사용하여 암호화됩니다.
- 클러스터가 삭제될 때 생성된 최종 스냅샷은 클러스터와 연결된 고객 관리형 키를 사용하여 암호화됩니다.
- 수동으로 생성한 스냅샷은 클러스터에 연결된 KMS 키를 사용하기 위해 기본적으로 암호화됩니다. 다른 고객 관리형 키를 선택하여 이를 재정의할 수 있습니다.
- 스냅샷 복사는 기본적으로 소스 스냅샷과 연결된 고객 관리형 키를 사용합니다. 다른 고객 관리형 키를 선택하여 이를 재정의할 수 있습니다.

> **Note**
>
> - 선택한 Amazon S3 버킷으로 스냅샷을 내보낼 때 고객 관리형 키를 사용할 수 없습니다. 그러나 Amazon S3으로 내보낸 모든 스냅샷은 서버 측 암호화를 사용하여 암호화됩니다. 스냅샷 파일을 새 S3 개체로 복사하고 고객 관리형 KMS 키를 사용하여 암호화하거나, KMS 키를 사용하여 기본 암호화로 설정된 다른 S3 버킷에 파일을 복사하거나, 파일 자체에서 암호화 옵션을 변경할 수 있습니다.
> - 또한 고객 관리형 키를 사용하여 암호화에 고객 관리형 키를 사용하지 않는 수동으로 생성한 스냅샷을 암호화할 수도 있습니다. 이 옵션을 사용하면 원래 클러스터에서 데이터가 암호화되지 않더라도 KMS 키를 사용하여 Amazon S3에 저장된 스냅샷 파일이 암호화됩니다.

스냅샷에서 복원하면 새 클러스터를 생성할 때 사용할 수 있는 암호화 옵션과 유사한 암호화 옵션을 선택할 수 있습니다.

- 클러스터를 암호화하는 데 사용한 키에 대해 키를 삭제하거나 키를 비활성화하고 그랜트를 취소하면 클러스터를 복구할 수 없게 됩니다. 즉, 하드웨어 장애 후 이를 수정하거나 복구할 수 없습니다. AWS KMS에서는 최소 7일 이상의 대기 기간이 지난 후에만 루트 키를 삭제합니다. 키를 삭제한 후 다른 고객 관리형 키를 사용하여 보관용 스냅샷을 생성할 수 있습니다.
- 자동 키 교체 기능은 AWS KMS 루트 키의 속성을 그대로 보존하기 때문에 키가 교체되더라도 MemoryDB 데이터에 대한 액세스 권한에는 아무런 영향도 끼치지 않습니다. 암호화된 MemoryDB 클러스터는 새로운 루트 키를 생성하거나 이전 키에 대한 모든 참조를 업데이트하는 수동 키 교체를 지원하지 않습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 고객 루트 키 교체를 참조하세요.
- KMS 키를 사용하여 MemoryDB 클러스터를 암호화하려면 클러스터당 1개의 그랜트가 필요합니다. 이 그랜트는 클러스터의 수명 동안 사용됩니다. 또한 스냅샷을 생성하는 동안 스냅샷당 하나의 권한 부여가 사용됩니다. 이 그랜트는 스냅샷이 생성되면사용 중지됩니다.
- AWS KMS 그랜트 및 제한에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서에서 할당량을 참조하세요.

참고

- MemoryDB에서 전송 중 데이터 암호화(TLS)
- MemoryDB 및 Amazon VPC
- MemoryDB의 ID 및 액세스 관리

MemoryDB에서 전송 중 데이터 암호화(TLS)

데이터를 안전하게 보관하기 위해 MemoryDB 및 Amazon EC2에서는 서버의 데이터에 대한 무단 액세스를 방지하는 메커니즘을 제공합니다. MemoryDB에서는 전송 중 데이터 암호화 기능을 제공해 한 위치에서 다른 위치로 데이터 이동 시 데이터를 보호할 수 있는 수단을 제공합니다. 예를 들어 클러스터 내 프라이머리 노드에서 읽기 전용 복제본 노드로 또는 클러스터와 애플리케이션 사이에서 데이터를 이동할 수 있습니다.

주제

- 전송 중 데이터 암호화 개요
- 다음 사항도 참조하세요.

전송 중 데이터 암호화 개요

MemoryDB 전송 중 암호화는 데이터가 한 위치에서 다른 위치로 전송 중일 때 가장 취약한 지점에서 데이터의 보안을 강화하는 기능입니다.

MemoryDB 전송 중 데이터 암호화는 다음 기능을 구현합니다.

- 암호화된 연결 - 서버와 클라이언트 연결이 둘 다 암호화된 전송 계층 보안(TLS)입니다.
- 암호화된 복제 - 기본 노드와 복제본 노드 간에 이동하는 데이터가 암호화됩니다.
- 서버 인증 - 클라이언트가 자신이 올바른 서버에 연결 중임을 인증할 수 있습니다.

2023년 7월 20일부터 TLS 1.2는 신규 및 기존 클러스터에 지원되는 최소 버전입니다. AWS에서 TLS 1.2에 대해 자세히 알아보려면 이 링크를 사용하세요.

MemoryDB 클러스터 연결에 대한 자세한 내용은 redis-cli를 사용하여 MemoryDB 노드에 연결 단원을 참조하세요.

다음 사항도 참조하세요.

- MemoryDB의 미사용 데이터 암호화
- 액세스 제어 목록(ACL)으로 사용자 인증
- MemoryDB 및 Amazon VPC
- MemoryDB의 ID 및 액세스 관리

액세스 제어 목록(ACL)을 사용하여 사용자 인증

액세스 제어 목록(ACL)을 사용하여 사용자를 인증할 수 있습니다.

ACL을 사용하면 사용자를 그룹화하여 클러스터 액세스를 제어할 수 있습니다. 이러한 액세스 제어 목록은 클러스터에 대한 액세스를 구성하는 방법으로 설계되었습니다.

ACL를 사용할 경우, 다음 섹션에 설명된 대로 액세스 문자열을 사용하여 사용자를 생성하고 특정 사용 권한을 할당합니다. 사용자를 특정 역할(관리자, 인사 관리)에 맞게 조정되어 있으며 하나 이상의

MemoryDB 클러스터에 배포되는 액세스 제어 목록에 할당합니다. 이렇게 하면 동일한 MemoryDB 클러스터 또는 클러스터를 사용하는 클라이언트 간에 보안 경계를 설정하고 클라이언트가 서로의 데이터에 액세스하지 못하게 할 수 있습니다.

ACL은 Redis OSS 6의 ACL 도입을 지원하도록 설계되었습니다. MemoryDB 클러스터에서 ACL을 사용할 때 몇 가지 제약 사항이 있습니다.

- 액세스 문자열에는 암호를 지정할 수 없습니다. CreateUser 또는 UpdateUser 직접 호출을 사용하여 암호를 설정합니다.
- 사용자 권한의 경우 on 및 `off`를 액세스 문자열의 일부로 전달합니다. 액세스 문자열에 둘 다 지정되지 않은 경우, 사용자에게 `off`이(가) 할당되며 클러스터에 대한 액세스 권한이 없습니다.
- 금지된 명령은 사용할 수 없습니다. 금지된 명령을 지정하면 예외가 발생합니다. 명령 전체 목록은 제한된 명령 단원을 참조하세요.
- `reset` 명령을 액세스 문자열의 일부로 사용할 수 없습니다. API 파라미터에 암호를 지정하고 MemoryDB가 암호를 관리합니다. 따라서 사용자의 모든 암호를 제거할 수 있는 `reset`을 사용할 수 없습니다.
- Redis OSS 6에 ACL LIST 명령이 도입되었습니다. 이 명령은 각 사용자에게 적용된 ACL 규칙과 함께 사용자 목록을 반환합니다. MemoryDB는 `ACL LIST` 명령을 지원하지만 Redis OSS와 마찬가지로 암호 해시에 대한 지원은 포함하지 않습니다. MemoryDB에서는 DescribeUsers 작업을 사용하여 액세스 문자열에 포함된 규칙을 비롯한 유사한 정보를 얻을 수 있습니다. 그러나 DescribeUsers는 사용자 암호를 검색하지 않습니다.

  MemoryDB에서 지원하는 다른 읽기 전용 명령에는 ACL WHOAMI, ACL USERS 및 ACL CAT가 포함됩니다. MemoryDB는 다른 쓰기 기반 ACL 명령을 지원하지 않습니다.

MemoryDB와 함께 ACL을 사용하는 방법은 다음에 자세히 설명되어 있습니다.

주제

액세스 문자열을 사용하여 권한 지정

MemoryDB 클러스터에 대한 권한을 지정하려면 액세스 문자열을 생성한 후 AWS CLI 또는 AWS Management 콘솔을(를) 사용하여 사용자에게 지정합니다.

액세스 문자열은 사용자에게 적용되는 공백으로 구분된 규칙의 목록으로 정의됩니다. 사용자가 실행할 수 있는 명령과 사용자가 작업할 수 있는 키를 정의합니다. 명령을 실행하기 위해서는 사용자가 실행될 명령과 명령에 의해 액세스되는 모든 키에 액세스할 수 있어야 합니다. 규칙은 왼쪽에서 오른쪽으로 누적되어 적용되며, 제공된 문자열에 중복 항목이 있는 경우, 제공된 문자열 대신 단순화된 문자열이 사용될 수 있습니다.

ACL 규칙의 구문에 대한 자세한 내용은 ACL 섹션을 참조하세요.

다음 예에서 액세스 문자열은 사용 가능한 모든 키와 명령에 액세스할 수 있는 활성 사용자를 나타냅니다.

```
on ~* &* +@all
```

액세스 문자열 구문은 다음과 같이 구분됩니다.

- on - 사용자가 활성 사용자입니다.
- `~*` - 사용 가능한 모든 키에 대한 액세스 권한을 부여합니다.
- `&*` - 모든 pubsub 채널에 대한 액세스 권한을 부여합니다.
- `+@all` - 사용 가능한 모든 명령에 대한 액세스 권한을 부여합니다.

이전 설정은 최소한의 제한적인 설정입니다. 이러한 설정을 수정하여 보안을 강화할 수 있습니다.

다음 예에서 액세스 문자열은 “app::” 키스페이스로 시작하는 키에 대한 읽기 액세스로 제한된 액세스 권한을 가진 사용자를 나타냅니다.

```
on ~app::* -@all +@read
```

사용자가 액세스할 수 있는 명령을 나열하여 이러한 권한을 세부적으로 조정할 수 있습니다.

+*command1* - 명령에 대한 사용자의 액세스는 *command1*로 제한됩니다.

`+@category` - 사용자의 액세스는 명령 범주로 제한됩니다.

사용자에게 액세스 문자열을 할당하는 방법에 대한 자세한 내용은 콘솔 및 CLI를 사용하여 사용자 및 액세스 제어 목록 생성 섹션을 참조하세요.

기존 워크로드를 MemoryDB로 마이그레이션하는 경우, `ACL LIST`을(를) 호출하여 사용자 및 암호 해시를 제외하는 방식으로 액세스 문자열을 검색할 수 있습니다.

벡터 검색 기능

벡터 검색의 경우, 모든 검색 명령은 `@search` 범주 및 기존 범주 `@read`, `@write`, `@fast`에 속하며 또한 `@slow`는 검색 명령을 포함하도록 업데이트되었습니다. 사용자에게 범주에 대한 액세스 권한이 없는 경우 사용자는 범주 내의 어떤 명령에도 액세스할 수 없습니다. 예를 들어 사용자가 `@search`에 대한 액세스 권한이 없는 경우, 사용자는 검색 관련 명령을 실행할 수 없습니다.

다음의 표는 검색 명령이 적절한 범주에 매핑되었음을 나타냅니다.

VSS 명령	@read	@write	@fast	@slow
`FT.CREATE`		Y	Y	
`FT.DROPINDEX`		Y	Y	
`FT.LIST`	Y			Y
`FT.INFO`	Y		Y	
`FT.SEARCH`	Y			Y
`FT.AGGREGATE`	Y			Y
`FT.PROFILE`	Y			Y
`FT.ALIASADD`		Y	Y	
`FT.ALIASDEL`		Y	Y	
`FT.ALIASUPDATE`		Y	Y	

VSS 명령	@read	@write	@fast	@slow
FT._ALIAS LIST	Y			Y
FT.EXPLAI N	Y		Y	
FT.EXPLAI NCLI	Y		Y	
FT.CONFIG	Y		Y	

MemoryDB용 클러스터에 ACL 적용

MemoryDB ACL을 사용하려면 다음 단계를 수행하세요.

1. 하나 이상의 사용자를 생성합니다.
2. ACL을 생성하고 사용자를 목록에 추가합니다.
3. 클러스터에 ACL을 할당합니다.

이러한 단계는 다음에 자세히 설명되어 있습니다.

주제

- 콘솔 및 CLI를 사용하여 사용자 및 액세스 제어 목록 생성
- 콘솔 및 CLI를 사용하여 액세스 제어 목록 관리
- 클러스터에 액세스 제어 목록 할당

콘솔 및 CLI를 사용하여 사용자 및 액세스 제어 목록 생성

ACL 사용자에 대한 사용자 정보는 사용자 이름 및 선택적으로 암호 및 액세스 문자열로 구성됩니다. 액세스 문자열은 키와 명령에 대한 권한 수준을 제공합니다. 이름은 사용자마다 고유하며 엔진에 전달됩니다.

제공하는 사용자 권한이 ACL의 의도된 목적에 적합한지 확인합니다. 예를 들어, `Administrators`라는 ACL을 생성하는 경우, 해당 그룹에 추가하는 모든 사용자는 키 및 명령에 대한 전체 액세스 권한으

로 설정된 액세스 문자열을 가져야 합니다. `e-commerce` ACL에 있는 사용자의 경우, 액세스 문자열을 읽기 전용 액세스로 설정할 수 있습니다.

MemoryDB는 `"default"` 사용자 이름을 사용하여 계정당 기본 사용자를 자동으로 구성합니다. ACL에 명시적으로 추가하지 않는 한 어떤 클러스터와도 연결되지 않습니다. 이 사용자는 수정하거나 삭제할 수 없습니다. 이 사용자는 이전 Redis OSS 버전의 기본 동작과의 호환성을 위해 만들어졌으며 모든 명령을 직접적으로 호출하고 모든 키에 액세스할 수 있는 액세스 문자열을 가지고 있습니다.

기본 사용자가 포함된 모든 계정에 대해 변경할 수 없는 "개방형 액세스" ACL이 생성됩니다. 이 ACL은 기본 사용자 한 명이 소속될 수 있는 유일한 ACL입니다. 클러스터를 생성할 때 클러스터와 연결할 ACL을 선택해야 합니다. 기본 사용자에게 "개방형 액세스" ACL을 적용할 수도 있지만, 업무상 필요에 따라 권한이 제한된 사용자를 사용하여 ACL을 생성하는 것이 좋습니다.

TLS가 활성화되지 않은 클러스터는 "개방형 액세스" ACL을 사용하여 개방형 인증을 제공해야 합니다.

사용자 없이 ACL을 생성할 수 있습니다. 빈 ACL은 클러스터에 액세스할 수 없으며 TLS 지원 클러스터와만 연결할 수 있습니다.

사용자를 생성할 때 최대 두 개의 암호를 설정할 수 있습니다. 암호를 수정해도 클러스터에 대한 모든 기존 연결이 유지됩니다.

특히 ACLs for MemoryDB을 사용할 때 이러한 사용자 암호 제약 조건에 유의해야 합니다.

- 암호는 16~128자 길이의 인쇄 가능한 문자여야 합니다.
- 영숫자가 아닌 다음과 같은 문자는 허용되지 않습니다. `,` `""` `/` `@`.

콘솔 및 CLI를 사용하여 사용자 관리

사용자 생성(콘솔)

콘솔에서 사용자를 생성하려면

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Users(사용자)를 선택합니다.
3. 사용자 생성(Create user)을 선택합니다.
4. 사용자 생성 페이지에서 이름을 입력합니다.

   클러스터 명명 제약 조건은 다음과 같습니다.

  - 1~40자의 영숫자 또는 하이픈으로 구성되어야 합니다.
  - 문자로 시작해야 합니다.
  - 하이픈 2개가 연속될 수 없습니다.
  - 끝에 하이픈이 올 수 없습니다.

5. 암호에서 최대 두 개의 암호를 입력할 수 있습니다.
6. 액세스 문자열에 액세스 문자열을 입력합니다. 액세스 문자열은 사용자에게 허용할 키와 명령에 대한 권한 수준을 설정합니다.
7. 태그의 경우, 선택적으로 태그를 적용하여 사용자를 검색 및 필터링하거나 AWS 비용을 추적할 수 있습니다.
8. 생성(Create)을 선택합니다.

AWS CLI을(를) 사용하여 사용자 생성

CLI를 사용하여 사용자를 생성하려면

- create-user 명령을 사용하여 사용자를 생성합니다.

  Linux, macOS, Unix의 경우:

```
aws memorydb create-user \
  --user-name user-name-1 \
  --access-string "~objects:* ~items:* ~public:*" \
  --authentication-mode \
        Passwords="abc",Type=password
```

  Windows의 경우:

```
aws memorydb create-user ^
  --user-name user-name-1 ^
  --access-string "~objects:* ~items:* ~public:*" ^
  --authentication-mode \
        Passwords="abc",Type=password
```

사용자 수정(콘솔)

콘솔에서 사용자를 수정하려면

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Users(사용자)를 선택합니다.
3. 수정할 사용자 옆의 라디오 단추를 선택하고 작업 -> 수정을 선택합니다.
4. 비밀번호를 수정하려면 암호 수정 라디오 버튼을 선택합니다. 단, 암호가 두 개인 경우, 둘 중 하나를 수정할 때는 둘 다 입력해야 합니다.
5. 액세스 문자열을 업데이트하려면 새 문자열을 입력하세요.
6. 수정을 선택합니다.

AWS CLI을(를) 사용하여 사용자 수정

CLI를 사용하여 사용자를 수정하려면

1. update-user 명령을 사용하여 사용자를 수정합니다.
2. 사용자를 수정하면 해당 사용자와 연결된 액세스 제어 목록이 업데이트되고 ACL과 연결된 클러스터도 업데이트됩니다. 기존 연결은 모두 유지됩니다. 예를 들면 다음과 같습니다.

   Linux, macOS, Unix의 경우:

   ```
   aws memorydb update-user \
     --user-name user-name-1 \
     --access-string "~objects:* ~items:* ~public:*"
   ```

   Windows의 경우:

   ```
   aws memorydb update-user ^
     --user-name user-name-1 ^
     --access-string "~objects:* ~items:* ~public:*"
   ```

사용자 세부 정보 보기(콘솔)

콘솔에서 사용자 세부 정보를 보려면

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Users(사용자)를 선택합니다.
3. 사용자 이름에서 사용자를 선택하거나 검색 상자를 사용하여 사용자를 찾습니다.
4. 사용자 설정에서 사용자의 액세스 문자열, 암호 수, 상태 및 Amazon 리소스 이름(ARN) 을 검토할 수 있습니다.
5. 액세스 제어 목록(ACL)에서 사용자가 속한 ACL을 검토할 수 있습니다.
6. 태그에서 사용자와 관련된 모든 태그를 검토할 수 있습니다.

AWS CLI을(를) 사용하여 사용자 세부 정보 보기

describe-users 명령을 사용하여 사용자의 세부 정보를 볼 수 있습니다.

```
aws memorydb describe-users \
  --user-name my-user-name
```

사용자 삭제(콘솔)

콘솔에서 사용자를 삭제하려면

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Users(사용자)를 선택합니다.
3. 수정하려는 사용자 옆의 라디오 단추를 선택하고 작업 -> 삭제를 선택합니다.
4. 확인하려면 확인 텍스트 상자에 `delete`을(를) 입력한 다음 삭제를 선택합니다.
5. 취소하려면 취소를 선택합니다.

AWS CLI을(를) 사용하여 사용자 삭제

CLI를 사용하여 사용자를 삭제하려면

- delete-user 명령을 사용하여 사용자를 삭제합니다.

계정이 삭제되고 해당 계정이 속한 모든 액세스 제어 목록에서 제거됩니다. 다음은 예입니다.

Linux, macOS, Unix의 경우:

```
aws memorydb delete-user \
  --user-name user-name-2
```

Windows의 경우:

```
aws memorydb delete-user ^
  --user-name user-name-2
```

콘솔 및 CLI를 사용하여 액세스 제어 목록 관리

다음과 같이 액세스 제어 목록을 생성하여 하나 이상의 클러스터에 대한 사용자 액세스를 구성하고 제어할 수 있습니다.

콘솔에서 다음 절차에 따라 액세스 제어 목록을 관리합니다.

액세스 제어 목록(ACL) 생성(콘솔)

콘솔을 사용하여 액세스 제어 목록을 생성하려면

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 액세스 제어 목록(ACL) 을 선택합니다.
3. ACL 생성을 선택합니다.
4. 액세스 제어 목록(ACL) 생성 페이지에서 ACL 이름을 입력합니다.

   클러스터 명명 제약 조건은 다음과 같습니다.

   - 1~40자의 영숫자 또는 하이픈으로 구성되어야 합니다.
   - 문자로 시작해야 합니다.
   - 하이픈 2개가 연속될 수 없습니다.
   - 끝에 하이픈이 올 수 없습니다.
5. 선택된 사용자에서 다음 작업 중 하나를 수행합니다.

   a. 새 사용자를 생성하려면 사용자 생성을 선택합니다.

b. 관리를 선택한 다음 사용자 관리 대화 상자에서 사용자를 선택한 다음 선택을 선택하여 사용자를 추가합니다.

6. 태그의 경우, 선택적으로 태그를 적용하여 ACL을 검색 및 필터링하거나 AWS 비용을 추적할 수 있습니다.
7. 생성(Create)을 선택합니다.

AWS CLI을 사용하여 액세스 제어 목록(ACL) 생성

다음 절차에 따라 CLI를 사용하여 액세스 제어 목록을 생성합니다.

CLI를 사용하여 새 ACL을 생성하고 사용자를 추가하려면

- create-acl 명령을 사용하여 ACL을 생성합니다.

  Linux, macOS, Unix의 경우:

```
aws memorydb create-acl \
  --acl-name "new-acl-1" \
  --user-names "user-name-1" "user-name-2"
```

  Windows의 경우:

```
aws memorydb create-acl ^
  --acl-name "new-acl-1" ^
  --user-names "user-name-1" "user-name-2"
```

액세스 제어 목록(ACL) 수정(콘솔)

콘솔을 사용하여 액세스 제어 목록을 수정하려면

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 액세스 제어 목록(ACL) 을 선택합니다.
3. 수정할 ACL을 선택하고 수정을 선택합니다.
4. 수정 페이지의 선택된 사용자에서 다음 중 하나를 수행합니다.

   a. ACL에 추가할 사용자 생성을 선택하여 새 사용자를 생성합니다.

   b. 관리를 선택한 다음 사용자 관리 대화 상자에서 사용자를 선택하거나 선택 취소한 다음 선택을 선택하여 사용자를 추가하거나 제거합니다.

5. 액세스 제어 목록(ACL) 생성 페이지에서 ACL 이름을 입력합니다.

   클러스터 명명 제약 조건은 다음과 같습니다.

   - 1~40자의 영숫자 또는 하이픈으로 구성되어야 합니다.
   - 문자로 시작해야 합니다.
   - 하이픈 2개가 연속될 수 없습니다.
   - 끝에 하이픈이 올 수 없습니다.

6. 선택된 사용자에서 다음 작업 중 하나를 수행합니다.

   a. 새 사용자를 생성하려면 사용자 생성을 선택합니다.

   b. 관리를 선택한 다음 사용자 관리 대화 상자에서 사용자를 선택한 다음 선택을 선택하여 사용자를 추가합니다.

7. 변경 내용을 저장하려면 수정을 선택하고 변경 내용을 삭제하려면 취소를 선택합니다.

AWS CLI을 사용하여 액세스 제어 목록(ACL) 수정

CLI를 사용하여 새 사용자를 추가하거나 현재 멤버를 제거하여 ACL을 수정하려면

- update-acl 명령을 사용하여 ACL을 수정합니다.

  Linux, macOS, Unix의 경우:

```
aws memorydb update-acl --acl-name new-acl-1 \
--user-names-to-add user-name-3 \
--user-names-to-remove user-name-2
```

  Windows의 경우:

```
aws memorydb update-acl --acl-name new-acl-1 ^
--user-names-to-add user-name-3 ^
--user-names-to-remove user-name-2
```

> **Note**
>
> ACL 에서 제거된 사용자에게 속하는 열려 있는 모든 연결이 이 명령으로 종료됩니다.

액세스 제어 목록(ACL) 세부 정보 보기(콘솔)

콘솔에서 ACL 세부 정보를 보려면

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 액세스 제어 목록(ACL)을 선택합니다.
3. ACL 이름에서 ACL을 선택하거나 검색 상자를 사용하여 ACL을 찾습니다.
4. 사용자에서 ACL과 관련된 사용자 목록을 검토할 수 있습니다.
5. 관련 클러스터에서 ACL이 속한 클러스터를 검토할 수 있습니다.
6. 태그에서 ACL과 관련된 모든 태그를 검토할 수 있습니다.

AWS CLI를 사용하여 액세스 제어 목록(ACL) 보기

describe-acls 명령을 사용하여 ACL의 세부 정보를 볼 수 있습니다.

```
aws memorydb describe-acls \
  --acl-name test-group
```

액세스 제어 목록(ACL) 삭제(콘솔)

콘솔을 사용하여 액세스 제어 목록을 삭제하려면

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 액세스 제어 목록(ACL)을 선택합니다.
3. 수정하려는 ACL을 선택한 다음 삭제를 선택합니다.
4. 삭제 페이지에서 확인 상자에 `delete`를 입력하고 삭제 또는 취소를 선택하여 ACL이 삭제되지 않도록 합니다.

그룹에 속한 사용자가 아닌 ACL 자체가 삭제됩니다.

다음을 사용하여 액세스 제어 목록(ACL) 삭제 AWS CLI

CLI를 사용하여 ACL을 삭제하려면

- delete-acl 명령을 사용하여 태그를 삭제합니다.

  Linux, macOS, Unix의 경우:

```
aws memorydb delete-acl /
   --acl-name
```

  Windows의 경우:

```
aws memorydb delete-acl ^
   --acl-name
```

  이전 예에서는 다음 응답을 반환합니다.

```
aws memorydb delete-acl --acl-name "new-acl-1"
{
    "ACLName": "new-acl-1",
    "Status": "deleting",
    "EngineVersion": "6.2",
    "UserNames": [
        "user-name-1",
        "user-name-3"
    ],
    "clusters": [],
    "ARN":"arn:aws:memorydb:us-east-1:493071037918:acl/new-acl-1"
}
```

클러스터에 액세스 제어 목록 할당

ACL을 생성하고 사용자를 추가한 후 ACL 구현의 마지막 단계는 ACL을 클러스터에 할당하는 것입니다.

콘솔을 사용하여 클러스터에 액세스 제어 목록 할당

AWS Management 콘솔을(를) 사용하여 클러스터에 ACL을 추가하려면 MemoryDB 클러스터 생성을 참조하세요.

AWS CLI을(를) 사용하여 클러스터에 액세스 제어 목록 할당

다음 AWS CLI 작업은 전송 중 데이터 암호화(TLS)가 활성화되고 acl-name 파라미터의 값이 *my-acl-name*인 클러스터를 생성합니다. 서브넷 그룹 `subnet-group`는 존재하는 서브넷 그룹으로 대체합니다.

키 파라미터

- **`--engine-version`** - 6.2가 되어야 합니다.
- **`--tls-enabled`** - 인증 및 ACL 연결에 사용됩니다.
- **`--acl-name`** - 이 값은 클러스터에 대해 지정된 액세스 권한을 가진 사용자로 구성된 액세스 제어 목록을 제공합니다.

Linux, macOS, Unix의 경우:

```
aws memorydb create-cluster \
    --cluster-name "new-cluster" \
    --description "new-cluster" \
    --engine-version "6.2" \
    --node-type db.r6g.large \
    --tls-enabled \
    --acl-name "new-acl-1" \
    --subnet-group-name "subnet-group"
```

Windows의 경우:

```
aws memorydb create-cluster ^
    --cluster-name "new-cluster" ^
    --cluster-description "new-cluster" ^
    --engine-version "6.2" ^
    --node-type db.r6g.large ^
    --tls-enabled ^
    --acl-name "new-acl-1" ^
    --subnet-group-name "subnet-group"
```

다음 AWS CLI 작업은 전송 중 데이터 암호화(TLS)가 활성화되고 acl-name 파라미터의 값이 `new-acl-2`인 클러스터를 수정합니다.

Linux, macOS, Unix의 경우:

```
aws memorydb update-cluster \
```

```
    --cluster-name cluster-1 \
    --acl-name "new-acl-2"
```

Windows의 경우:

```
aws memorydb update-cluster ^
    --cluster-name cluster-1 ^
    --acl-name "new-acl-2"
```

IAM을 통한 인증

주제

- 개요
- 제한 사항
- 설정
- Connecting

개요

클러스터가 Valkey 또는 Redis OSS 버전 7 이상을 사용하도록 구성된 경우 IAM 인증을 사용하여 AWSIAM ID를 사용하여 MemoryDB에 대한 연결을 인증할 수 있습니다. 그러면 보안 모델이 강화되고 여러 보안 관리 작업이 단순화됩니다. IAM 인증을 사용하면 각 개별 MemoryDB 클러스터 및 MemoryDB 사용자별로 액세스 제어를 세분화해 구성하고 최소 권한 원칙을 고수할 수 있습니다. MemoryDB용 IAM 인증은 `AUTH` 또는 `HELLO` 명령 내에서 장수명 MemoryDB 사용자 암호 대신, 단수명 IAM 인증 토큰을 제공하여 작동합니다. IAM 인증 토큰에 대한 자세한 내용은 AWS일반 참조 안내서의 서명 버전 4 서명 프로세스와 아래 코드 예제를 참조하세요.

IAM ID 및 관련 정책을 사용하여 Valkey 또는 Redis OSS 액세스를 더욱 엄격히 제한할 수도 있습니다. 페더레이션 자격 증명 공급자로부터 제공받은 사용자에게 바로 MemoryDB 클러스터 액세스를 부여할 수도 있습니다.

MemoryDB에서 AWSIAM을 사용하려면 먼저 인증 모드가 IAM으로 설정된 MemoryDB 사용자를 생성한 다음 IAM 자격 증명을 생성하거나 재사용할 수 있습니다. IAM 자격 증명에 관련 정책이 있어야 MemoryDB 클러스터와 MemoryDB 사용자에게 `memorydb:Connect` 작업을 부여할 수 있습니다. 구성이 완료되면 IAM 사용자 또는 역할의 자격 AWS증명을 사용하여 IAM 인증 토큰을 생성할 수 있습니다. 마지막으로, MemoryDB 클러스터 노드에 연결할 때 Valkey 또는 Redis OSS 클라이언트에서 암호

로 단수명 IAM 인증 토큰을 제공해야 합니다. 자격 증명 공급자를 지원하는 클라이언트는 각각의 새 연결에 자동으로 임시 자격 증명을 자동 생성할 수 있습니다. MemoryDB는 IAM가 활성화된 MemoryDB 사용자의 연결 요청에 IAM 인증을 수행하고 IAM과의 연결 요청을 검증합니다.

제한 사항

IAM 데이터베이스 인증을 사용할 때 다음 제한이 적용됩니다.

- IAM 인증은 Valkey 또는 Redis OSS 엔진 버전 7.0 이상에 사용할 수 있습니다.
- IAM 인증 토큰은 15분간 유효합니다. 장수명 연결의 경우, 자격 증명 공급자 인터페이스를 지원하는 Redis OSS 클라이언트를 사용하는 것이 좋습니다.
- MemoryDB로의 IAM 인증 연결은 12시간 후에 자동으로 연결이 해제됩니다. `AUTH` 또는 `HELLO` 명령과 새 IAM 인증 토큰을 전송하여 연결을 12시간 연장할 수 있습니다.
- IAM 인증은 `MULTI EXEC` 명령에서 지원되지 않습니다.
- 현재 IAM 인증은 모든 전역 조건 컨텍스트 키를 지원하지 않습니다. 전역 조건 컨텍스트 키에 대한 자세한 내용은 IAM 사용 설명서의 [AWS 전역 조건 컨텍스트 키]를 참조하세요.

설정

IAM 인증을 설정하려면:

1. 클러스터 생성

```
aws memorydb create-cluster \
    --cluster-name cluster-01 \
    --description "MemoryDB IAM auth application"
    --node-type db.r6g.large \
    --engine-version 7.0 \
    --acl-name open-access
```

2. 아래에서와 같이, 역할에 IAM 신뢰 정책 문서를 생성하여 계정이 새 역할을 수임할 수 있도록 합니다. 정책을 trust-policy.json 파일에 저장합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
```

```
        "Principal": { "AWS": "arn:aws:iam::123456789012:root" },
        "Action": "sts:AssumeRole"
    }
}
```

3. 아래에서와 같이, IAM 정책 문서를 생성합니다. 정책을 policy.json 파일에 저장합니다.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect" : "Allow",
      "Action" : [
        "memorydb:connect"
      ],
      "Resource" : [
        "arn:aws:memorydb:us-east-1:123456789012:cluster/cluster-01",
        "arn:aws:memorydb:us-east-1:123456789012:user/iam-user-01"
      ]
    }
  ]
}
```

4. IAM 역할을 생성합니다.

```
aws iam create-role \
  --role-name "memorydb-iam-auth-app" \
  --assume-role-policy-document file://trust-policy.json
```

5. IAM 정책을 생성합니다.

```
aws iam create-policy \
  --policy-name "memorydb-allow-all" \
  --policy-document file://policy.json
```

6. IAM 정책을 역할에 연결합니다.

```
aws iam attach-role-policy \
 --role-name "memorydb-iam-auth-app" \
 --policy-arn "arn:aws:iam::123456789012:policy/memorydb-allow-all"
```

7. IAM가 활성화된 사용자를 새로 생성합니다.

```
aws memorydb create-user \
  --user-name iam-user-01 \
  --authentication-mode Type=iam \
  --access-string "on ~* +@all"
```

8. ACL을 생성하고 사용자를 연결합니다.

```
aws memorydb create-acl \
  --acl-name iam-acl-01 \
  --user-names iam-user-01

aws memorydb update-cluster \
  --cluster-name cluster-01 \
  --acl-name iam-acl-01
```

Connecting

토큰을 암호로 연결

먼저 AWS SigV4 사전 서명된 요청을 사용하여 단수명 IAM 인증 토큰을 생성해야 합니다. 그런 다음, MemoryDB 클러스터에 연결할 때 아래 예제에서와 같이 IAM 인증 토큰을 암호로 제공합니다.

```
String userName = "insert user name"
String clusterName = "insert cluster name"
String region = "insert region"

// Create a defaultAWSCredentials provider.
// This will look forAWScredentials defined in environment variables or system
 properties.
AWSCredentialsProvider awsCredentialsProvider = new
 DefaultAWSCredentialsProviderChain();

// Create an IAM authentication token request and signed it using theAWScredentials.
// The pre-signed request URL is used as an IAM authentication token for MemoryDB.
IAMAuthTokenRequest iamAuthTokenRequest = new IAMAuthTokenRequest(userName,
 clusterName, region);
String iamAuthToken =
 iamAuthTokenRequest.toSignedRequestUri(awsCredentialsProvider.getCredentials());

// Construct URL with IAM Auth credentials provider
```

```
RedisURI redisURI = RedisURI.builder()
    .withHost(host)
    .withPort(port)
    .withSsl(ssl)
    .withAuthentication(userName, iamAuthToken)
    .build();

// Create a new Lettuce client
RedisClusterClient client = RedisClusterClient.create(redisURI);
client.connect();
```

아래는 IAMAuthTokenRequest의 정의입니다.

```
public class IAMAuthTokenRequest {
    private static final HttpMethodName REQUEST_METHOD = HttpMethodName.GET;
    private static final String REQUEST_PROTOCOL = "http://";
    private static final String PARAM_ACTION = "Action";
    private static final String PARAM_USER = "User";
    private static final String ACTION_NAME = "connect";
    private static final String SERVICE_NAME = "memorydb";
    private static final long TOKEN_EXPIRY_SECONDS = 900;

    private final String userName;
    private final String clusterName;
    private final String region;

    public IAMAuthTokenRequest(String userName, String clusterName, String region) {
        this.userName = userName;
        this.clusterName = clusterName;
        this.region = region;
    }

    public String toSignedRequestUri(AWSCredentials credentials) throws
 URISyntaxException {
        Request<Void> request = getSignableRequest();
        sign(request, credentials);
        return new URIBuilder(request.getEndpoint())
            .addParameters(toNamedValuePair(request.getParameters()))
            .build()
            .toString()
            .replace(REQUEST_PROTOCOL, "");
    }
```

```
    private <T> Request<T> getSignableRequest() {
        Request<T> request  = new DefaultRequest<>(SERVICE_NAME);
        request.setHttpMethod(REQUEST_METHOD);
        request.setEndpoint(getRequestUri());
        request.addParameters(PARAM_ACTION, Collections.singletonList(ACTION_NAME));
        request.addParameters(PARAM_USER, Collections.singletonList(userName));
        return request;
    }

    private URI getRequestUri() {
        return URI.create(String.format("%s%s/", REQUEST_PROTOCOL, clusterName));
    }

    private <T> void sign(SignableRequest<T> request, AWSCredentials credentials) {
        AWS4Signer signer = new AWS4Signer();
        signer.setRegionName(region);
        signer.setServiceName(SERVICE_NAME);

        DateTime dateTime = DateTime.now();
        dateTime = dateTime.plus(Duration.standardSeconds(TOKEN_EXPIRY_SECONDS));

        signer.presignRequest(request, credentials, dateTime.toDate());
    }

    private static List<NameValuePair> toNamedValuePair(Map<String, List<String>> in) {
        return in.entrySet().stream()
            .map(e -> new BasicNameValuePair(e.getKey(), e.getValue().get(0)))
            .collect(Collectors.toList());
    }
}
```

보안 인증 정보 공급자와 연결

아래 코드는 IAM 인증 보안 인증 정보 공급자를 사용하여 MemoryDB로 인증하는 방법을 보여줍니다.

```
String userName = "insert user name"
String clusterName = "insert cluster name"
String region = "insert region"

// Create a defaultAWSCredentials provider.
// This will look forAWScredentials defined in environment variables or system
 properties.
AWSCredentialsProvider awsCredentialsProvider = new
 DefaultAWSCredentialsProviderChain();
```

```
// Create an IAM authentication token request. Once this request is signed it can be
 used as an
// IAM authentication token for MemoryDB.
IAMAuthTokenRequest iamAuthTokenRequest = new IAMAuthTokenRequest(userName,
 clusterName, region);

// Create a credentials provider using IAM credentials.
RedisCredentialsProvider redisCredentialsProvider = new
 RedisIAMAuthCredentialsProvider(
    userName, iamAuthTokenRequest, awsCredentialsProvider);

// Construct URL with IAM Auth credentials provider
RedisURI redisURI = RedisURI.builder()
    .withHost(host)
    .withPort(port)
    .withSsl(ssl)
    .withAuthentication(redisCredentialsProvider)
    .build();

// Create a new Lettuce cluster client
RedisClusterClient client = RedisClusterClient.create(redisURI);
client.connect();
```

다음은 자격 증명 공급자에 IAMAuthTokenRequest를 래핑하여 필요할 때 임시 자격 증명을 자동 생성하는 Lettuce 클러스터 클라이언트의 예입니다.

```
public class RedisIAMAuthCredentialsProvider implements RedisCredentialsProvider {
    private static final long TOKEN_EXPIRY_SECONDS = 900;

    private final AWSCredentialsProvider awsCredentialsProvider;
    private final String userName;
    private final IAMAuthTokenRequest iamAuthTokenRequest;
    private final Supplier<String> iamAuthTokenSupplier;

    public RedisIAMAuthCredentialsProvider(String userName,
        IAMAuthTokenRequest iamAuthTokenRequest,
        AWSCredentialsProvider awsCredentialsProvider) {
        this.userName = userName;
        this.awsCredentialsProvider = awsCredentialsProvider;
        this.iamAuthTokenRequest = iamAuthTokenRequest;
```

```
        this.iamAuthTokenSupplier =
 Suppliers.memoizeWithExpiration(this::getIamAuthToken, TOKEN_EXPIRY_SECONDS,
 TimeUnit.SECONDS);
    }

    @Override
    public Mono<RedisCredentials> resolveCredentials() {
        return Mono.just(RedisCredentials.just(userName, iamAuthTokenSupplier.get()));
    }

    private String getIamAuthToken() {
        return
 iamAuthTokenRequest.toSignedRequestUri(awsCredentialsProvider.getCredentials());
    }
```

MemoryDB의 ID 및 액세스 관리

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어하는 데 도움이 AWS 서비스 되는 입니다. IAM 관리자는 누가 MemoryDB 리소스를 사용하도록 인증되고(로그인됨) 권한이 부여되는지(권한 있음)를 제어합니다. IAM은 추가 비용 없이 사용할 수 AWS 서비스 있는 입니다.

주제

대상

AWS Identity and Access Management (IAM)를 사용하는 방법은 역할에 따라 다릅니다.

- 서비스 사용자 - 기능에 액세스할 수 없는 경우 관리자에게 권한 요청(참조MemoryDB ID 및 액세스 문제 해결)
- 서비스 관리자 - 사용자 액세스 결정 및 권한 요청 제출(MemoryDB에서 IAM을 사용하는 방식 참조)
- IAM 관리자 - 액세스를 관리하기 위한 정책 작성(MemoryDB 보안 인증 기반 정책 예제 참조)

ID를 통한 인증

인증은 자격 증명 자격 증명을 AWS 사용하여에 로그인하는 방법입니다. AWS 계정 루트 사용자, IAM 사용자 또는 IAM 역할을 수임하여 인증되어야 합니다.

AWS IAM Identity Center (IAM Identity Center), Single Sign-On 인증 또는 Google/Facebook 자격 증명과 같은 자격 증명 소스의 자격 증명을 사용하여 페더레이션 자격 증명으로 로그인할 수 있습니다. 로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의 AWS 계정에 로그인하는 방법 섹션을 참조하세요.

프로그래밍 방식 액세스를 위해는 요청에 암호화 방식으로 서명할 수 있는 SDK 및 CLI를 AWS 제공합니다. 자세한 내용은 IAM 사용자 설명서의 API 요청용AWS Signature Version 4 섹션을 참조하세요.

AWS 계정 루트 사용자

를 생성할 때 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 AWS 계정 theroot 사용자라는 하나의 로그인 자격 증명으로 AWS 계정시작합니다. 일상적인 태스크에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명이 필요한 작업은 IAM 사용자 설명서의 루트 사용자 자격 증명이 필요한 작업 섹션을 참조하세요.

페더레이션 ID

가장 좋은 방법은 인간 사용자에게 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 AWS 서비스 사용하여에 액세스하도록 요구하는 것입니다.

페더레이션 자격 증명은 엔터프라이즈 디렉터리, 웹 자격 증명 공급자 또는 자격 증명 소스의 자격 증명을 AWS 서비스 사용하여 Directory Service 에 액세스하는 사용자입니다. 페더레이션 ID는 임시 자격 증명을 제공하는 역할을 수임합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center를 추천합니다. 자세한 정보는 AWS IAM Identity Center 사용 설명서의 What is IAM Identity Center?를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가진 ID입니다. 장기 자격 증명이 있는 IAM 사용자 대신 임시 자격 증명을 사용하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 자격 증명 공급자와의 연동을 사용하여 임시 자격 증명을 AWS 사용하여에 액세스하도록 인간 사용자에게 요구하기를 참조하세요.

IAM 그룹은 IAM 사용자 모음을 지정하고 대규모 사용자 집합에 대한 관리 권한을 더 쉽게 만듭니다. 자세한 내용은 IAM 사용자 설명서의 IAM 사용자 사용 사례 섹션을 참조하세요.

IAM 역할

IAM 역할은 임시 자격 증명을 제공하는 특정 권한이 있는 자격 증명입니다. 사용자에서 IAM 역할(콘솔)로 전환하거나 또는 API 작업을 호출하여 역할을 수임할 수 있습니다. AWS CLI AWS 자세한 내용은 IAM 사용자 설명서의 역할 수임 방법을 참조하세요.

IAM 역할은 페더레이션 사용자 액세스, 임시 IAM 사용자 권한, 교차 계정 액세스, 교차 서비스 액세스 및 Amazon EC2에서 실행되는 애플리케이션에 유용합니다. 자세한 내용은 IAM 사용자 설명서의 교차 계정 리소스 액세스를 참조하세요.

정책을 사용하여 액세스 관리

정책을 AWS 생성하고 자격 증명 또는 리소스에 연결하여 AWS 에서 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결될 때 권한을 정의합니다.는 보안 주체가 요청할 때 이러한 정책을 AWS 평가합니다. 대부분의 정책은에 JSON 문서 AWS 로 저장됩니다. JSON 정책 문서에 대한 자세한 내용은 IAM 사용자 설명서의 JSON 정책 개요 섹션을 참조하세요.

정책을 사용하여 관리자는 어떤 보안 주체가 어떤 리소스에 대해 어떤 조건에서 작업을 수행할 수 있는지 정의하여 누가 무엇을 액세스할 수 있는지 지정합니다.

기본적으로 사용자 및 역할에는 어떠한 권한도 없습니다. IAM 관리자는 IAM 정책을 생성하고 사용자가 수임할 수 있는 역할에 추가합니다. IAM 정책은 작업을 수행하기 위해 사용하는 방법과 관계없이 작업에 대한 권한을 정의합니다.

자격 증명 기반 정책

ID 기반 정책은 ID(사용자, 사용자 그룹 또는 역할)에 연결하는 JSON 권한 정책 문서입니다. 이러한 정책은 자격 증명이 수행할 수 있는 작업, 대상 리소스 및 이에 관한 조건을 제어합니다. ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용자 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

ID 기반 정책은 인라인 정책(단일 ID에 직접 포함) 또는 관리형 정책(여러 ID에 연결된 독립 실행형 정책)일 수 있습니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용자 설명서의 관리형 정책 및 인라인 정책 중에서 선택 섹션을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 예를 들어 IAM 역할 신뢰 정책 및 Amazon S3 버킷 정책이 있습니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS 관리형 정책을 사용할 수 없습니다.

기타 정책 유형

AWS 는 보다 일반적인 정책 유형에서 부여한 최대 권한을 설정할 수 있는 추가 정책 유형을 지원합니다.

- 권한 경계 - ID 기반 정책에서 IAM 엔터티에 부여할 수 있는 최대 권한을 설정합니다. 자세한 정보는 IAM 사용자 설명서의 IAM 엔터티의 권한 범위를 참조하세요.
- 서비스 제어 정책(SCP) - AWS Organizations내 조직 또는 조직 단위에 대한 최대 권한을 지정합니다. 자세한 내용은AWS Organizations 사용 설명서의 서비스 제어 정책을 참조하세요.
- 리소스 제어 정책(RCP) – 계정의 리소스에 사용할 수 있는 최대 권한을 설정합니다. 자세한 내용은 AWS Organizations 사용 설명서의 리소스 제어 정책(RCP)을 참조하세요.
- 세션 정책 – 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 자세한 내용은 IAM 사용자 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 에서 여러 정책 유형이 관련될 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

MemoryDB에서 IAM을 사용하는 방식

IAM을 사용하여 MemoryDB에 대한 액세스를 관리하기 전에 MemoryDB와 함께 사용할 수 있는 IAM 기능을 알아보세요.

MemoryDB에서 사용할 수 있는 IAM 기능

IAM 특성	MemoryDB 지원
자격 증명 기반 정책	예
리소스 기반 정책	아니요
정책 작업	예
정책 리소스	예
정책 조건 키	예
ACL	예
ABAC(정책의 태그)	예
임시 보안 인증	예
엔터티 권한	예
서비스 역할	예
서비스 연결 역할	예

MemoryDB 및 기타 AWS 서비스가 대부분의 IAM 기능과 작동하는 방식을 개괄적으로 알아보려면 IAM 사용 설명서의 AWS IAM으로 작업하는 서비스를 참조하세요.

MemoryDB에 대한 보안 인증 기반 정책

자격 증명 기반 정책 지원: 예

자격 증명 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자 및 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

IAM 자격 증명 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. JSON 정책에서 사용할 수 있는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

MemoryDB 보안 인증 기반 정책 예제

MemoryDB 보안 인증 기반 정책의 예를 보려면 MemoryDB 보안 인증 기반 정책 예제 단원을 참조하세요.

MemoryDB 내 리소스 기반 정책

리소스 기반 정책 지원: 아니요

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예제는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는가 포함될 수 있습니다 AWS 서비스.

교차 계정 액세스를 활성화하려는 경우, 전체 계정이나 다른 계정의 IAM 개체를 리소스 기반 정책의 보안 주체로 지정할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM에서 교차 계정 리소스 액세스를 참조하세요.

MemoryDB의 정책 작업

정책 작업 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 `Action`요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다. 연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하세요.

MemoryDB 작업 목록을 보려면 서비스 승인 참조의 MemoryDB에서 정의한 작업을 참조하세요.

MemoryDB의 정책 작업은 작업 앞에 다음 접두사를 사용합니다.

```
MemoryDB
```

단일 문에서 여러 작업을 지정하려면 쉼표로 구분합니다.

```
"Action": [
      "MemoryDB:action1",
```

```
        "MemoryDB:action2"
            ]
```

와일드카드(*)를 사용하여 여러 작업을 지정할 수 있습니다. 예를 들어, `Describe`라는 단어로 시작하는 모든 작업을 지정하려면 다음 작업을 포함합니다.

```
"Action": "MemoryDB:Describe*"
```

MemoryDB 보안 인증 기반 정책의 예를 보려면 MemoryDB 보안 인증 기반 정책 예제 단원을 참조하세요.

MemoryDB의 정책 리소스

정책 리소스 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Resource` JSON 정책 요소는 작업이 적용되는 하나 이상의 객체를 지정합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

MemoryDB 리소스 유형 및 해당 ARN의 목록을 보려면 서비스 승인 참조의 MemoryDB에서 정의한 리소스를 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 MemoryDB에서 정의한 작업을 참조하세요.

MemoryDB 보안 인증 기반 정책의 예를 보려면 MemoryDB 보안 인증 기반 정책 예제 단원을 참조하세요.

MemoryDB의 정책 조건 키

서비스별 정책 조건 키 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

Condition 요소는 정의된 기준에 따라 문이 실행되는 시기를 지정합니다. 같음(equals) 또는 미만 (less than)과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

MemoryDB 보안 인증 기반 정책의 예를 보려면 MemoryDB 보안 인증 기반 정책 예제 단원을 참조하세요.

조건 키 사용

IAM 정책이 적용되는 방식을 결정하는 조건을 지정할 수 있습니다. MemoryDB에서 JSON 정책의 Condition 요소를 사용하여 요청 컨텍스트의 키를 정책에서 지정한 키 값과 비교할 수 있습니다. 자세한 정보는 IAM JSON 정책 요소: 조건을 참조하세요.

MemoryDB 조건 키 목록을 보려면 서비스 승인 참조의 MemoryDB의 조건 키를 참조하세요.

글로벌 조건 키의 목록은 AWS 글로벌 조건 컨텍스트 키를 참조하세요.

조건 지정: 조건 키 사용

세분화된 제어를 구현하려는 경우 특정 요청에 대한 개별 파라미터 집합을 제어하는 조건을 지정하는 IAM 권한 정책을 작성할 수 있습니다. 그런 다음 IAM 콘솔을 사용하여 만드는 IAM 사용자, 그룹 또는 역할에 정책을 적용할 수 있습니다.

조건을 적용하려면 IAM 정책 설명에 조건 정보를 추가합니다. 예를 들어 TLS가 비활성화된 MemoryDB 클러스터의 생성을 허용하지 않으려면 정책 문에 다음 조건을 지정할 수 있습니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "memorydb:CreateCluster"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
```

```
          "Bool": {
            "memorydb:TLSEnabled": "false"
          }
        }
      }
    ]
  }
```

태그 지정에 대한 자세한 내용은 [MemoryDB 리소스 태그 지정](#) 섹션을 참조하세요.

정책 조건 연산자 사용에 대한 자세한 내용은 [MemoryDB API 권한: 작업, 리소스 및 조건 참조](#) 섹션을 참조하세요.

정책 예: 조건을 사용하여 세부적인 파라미터 제어 구현

이 섹션에서는 이전에 나열된 MemoryDB 파라미터에 대한 세분화된 액세스 제어를 구현하기 위한 정책 예를 보여 줍니다.

1. memorydb:TLSEnabled - TLS가 활성화된 상태에서만 클러스터를 생성하도록 지정합니다.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
              {
            "Effect": "Allow",
            "Action": [
                "memorydb:CreateCluster"
            ],
            "Resource": [
                "arn:aws:memorydb:*:*:parametergroup/*",
                "arn:aws:memorydb:*:*:subnetgroup/*",
                "arn:aws:memorydb:*:*:acl/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "memorydb:CreateCluster"
            ],
            "Resource": [
                "*"
```

```
            ],
            "Condition": {
                "Bool": {
                    "memorydb:TLSEnabled": "true"
                }
            }
        }
    ]
}
```

2. memorydb:UserAuthenticationMode: - 특정 유형의 인증 모드(예: IAM)로 사용자를 생성할 수 있도록 지정합니다.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "memorydb:Createuser"
            ],
            "Resource": [
                "arn:aws:memorydb:*:*:user/*"
            ],
            "Condition": {
                "StringEquals": {
                    "memorydb:UserAuthenticationMode": "iam"
                }
            }
        }
    ]
}
```

   '거부' 기반 정책을 설정하는 경우 대소문자에 관계없이 특정 사용자 인증 모드 유형의 모든 직접 호출을 방지하려면 StringEqualsIgnoreCase 연산자를 사용하는 것이 좋습니다.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Deny",
      "Action": [
        "memorydb:CreateUser"
      ],
      "Resource": "*",
      "Condition": {
        "StringEqualsIgnoreCase": {
          "memorydb:UserAuthenticationMode": "password"
        }
      }
    }
  ]
}
```

MemoryDB의 액세스 제어 목록(ACL)

ACL 지원: 예

액세스 제어 목록(ACL)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

MemoryDB에서 속성 기반 액세스 제어(ABAC)

ABAC 지원(정책의 태그): 예

속성 기반 액세스 제어(ABAC)는 태그라고 불리는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. IAM 엔터티 및 AWS 리소스에 태그를 연결한 다음 보안 주체의 태그가 리소스의 태그와 일치할 때 작업을 허용하는 ABAC 정책을 설계할 수 있습니다.

태그에 근거하여 액세스를 제어하려면 `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`또는 `aws:TagKeys`조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다.

서비스가 모든 리소스 유형에 대해 세 가지 조건 키를 모두 지원하는 경우, 값은 서비스에 대해 예입니다. 서비스가 일부 리소스 유형에 대해서만 세 가지 조건 키를 모두 지원하는 경우, 값은 부분적입니다.

ABAC에 대한 자세한 내용은 IAM 사용자 설명서의 ABAC 권한 부여를 통한 권한 정의를 참조하세요. ABAC 설정 단계가 포함된 자습서를 보려면 IAM 사용자 설명서의 속성 기반 액세스 제어(ABAC) 사용을 참조하세요.

MemoryDB에서 임시 보안 인증 정보 사용

임시 자격 증명 지원: 예

임시 자격 증명은 AWS 리소스에 대한 단기 액세스를 제공하며 페더레이션을 사용하거나 역할을 전환할 때 자동으로 생성됩니다. 장기 액세스 키를 사용하는 대신 임시 자격 증명을 동적으로 생성하는 것이 AWS 좋습니다. 자세한 내용은 IAM 사용 설명서의 [IAM의 임시 보안 자격 증명] 및 [IAM으로 작업하는AWS 서비스] 섹션을 참조하세요.

MemoryDB의 서비스 간 보안 주체 권한

전달 액세스 세션(FAS) 지원: 예

전달 액세스 세션(FAS)은를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대한 요청과 AWS 서비스함께 사용합니다. FAS 요청 시 정책 세부 정보는 [전달 액세스 세션]을 참조하세요.

MemoryDB의 서비스 역할

서비스 역할 지원: 예

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 [IAM 역할]입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 [AWS 서비스 AWS에 권한을 위임할 역할 생성]을 참조하세요.

> **Warning**
>
> 서비스 역할에 대한 권한을 변경하면 MemoryDB 기능이 중단될 수 있습니다. MemoryDB에서 관련 지침을 제공하는 경우에만 서비스 역할을 편집하세요.

MemoryDB에 대한 서비스 연결 역할

서비스 연결 역할 지원: 예

서비스 연결 역할은에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은에 표시 AWS 계정 되며 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

서비스 연결 역할 생성 또는 관리에 대한 자세한 내용은 [IAM으로 작업하는AWS 서비스]를 참조하세요. 서비스 연결 역할 열에서 Yes가 포함된 서비스를 테이블에서 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예(Yes) 링크를 선택합니다.

MemoryDB 보안 인증 기반 정책 예제

기본적으로 사용자 및 역할에는 MemoryDB 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성(콘솔)을 참조하세요.

각 리소스 유형의 ARN 형식을 포함하여 MemoryDB에서 정의한 작업 및 리소스 유형에 대한 자세한 내용은 서비스 승인 참조의 MemoryDB에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

주제

- 정책 모범 사례
- MemoryDB 콘솔 사용
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용

정책 모범 사례

보안 인증 기반 정책에 따라 계정에서 사용자가 MemoryDB 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. 자격 증명 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따르세요.

- AWS 관리형 정책을 시작하고 최소 권한으로 전환 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반적인 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. 에서 사용할 수 있습니다 AWS 계정. 사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 추가로 줄이는 것이 좋습니다. 자세한 내용은 IAM 사용자 설명서의 AWS 관리형 정책 또는 AWS 직무에 대한 관리형 정책을 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우, 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용자 설명서에 있는 IAM의 정책 및 권한을 참조하세요.
- IAM 정책의 조건을 사용하여 액세스 추가 제한 – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어, SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. AWS 서비스와 같은 특정를 통해 사용되는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다 CloudFormation. 자세한 내용은 IAM 사용자 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.

- IAM Access Analyzer를 통해 IAM 정책을 확인하여 안전하고 기능적인 권한 보장 - IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 새로운 및 기존 정책을 확인합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 내용은 IAM 사용자 설명서의 IAM Access Analyzer에서 정책 검증을 참조하세요.
- 다중 인증(MFA) 필요 -에서 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 AWS 계정켭니다. API 작업을 직접적으로 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 내용은 IAM 사용자 설명서의 MFA를 통한 보안 API 액세스를 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용자 설명서의 IAM의 보안 모범 사례를 참조하세요.

MemoryDB 콘솔 사용

MemoryDB 콘솔에 액세스하려면 최소 권한 세트가 있어야 합니다. 이러한 권한은에서 MemoryDB 리소스에 대한 세부 정보를 나열하고 볼 수 있도록 허용해야 합니다 AWS 계정. 최소 필수 권한보다 더 제한적인 자격 증명 기반 정책을 생성하는 경우, 콘솔이 해당 정책에 연결된 엔티티(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

AWS CLI 또는 AWS API만 호출하는 사용자에게 최소 콘솔 권한을 허용할 필요는 없습니다. 대신, 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

사용자와 역할이 MemoryDB 콘솔을 계속 사용할 수 있도록 하려면 MemoryDB `ConsoleAccess` 또는 `ReadOnly` AWS 관리형 정책도 엔터티에 연결합니다. 자세한 내용은 IAM 사용 설명서의 사용자에게 권한 추가를 참조하세요.

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예제는 IAM 사용자가 자신의 사용자 ID에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 AWS CLI 또는 AWS API를 사용하여 프로그래밍 방식으로이 작업을 완료할 수 있는 권한이 포함됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
```

```
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

MemoryDB ID 및 액세스 문제 해결

다음 정보를 사용하여 MemoryDB 및 IAM에서 발생할 수 있는 공통적인 문제를 진단하고 수정할 수 있습니다.

주제

- [MemoryDB에서 작업을 수행할 권한이 없음]
- [iam:PassRole을 수행하도록 인증되지 않음]
- [내 AWS 계정 외부의 사람이 내 MemoryDB 리소스에 액세스하도록 허용하고 싶습니다.]

MemoryDB에서 작업을 수행할 권한이 없음

에서 작업을 수행할 권한이 없다는 AWS Management 콘솔 메시지가 표시되면 관리자에게 문의하여 도움을 받아야 합니다. 관리자는 사용자 이름과 암호를 제공한 사람입니다.

다음 예제 오류는 `mateojackson` 사용자가 콘솔을 사용하여 가상 *`my-example-widget`* 리소스에 대한 세부 정보를 보려고 하지만 가상 `MemoryDB:`*`GetWidget`* 권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 MemoryDB:GetWidget on resource: my-example-widget
```

이 경우 Mateo는 *`my-example-widget`* 작업을 사용하여 `MemoryDB:`*`GetWidget`* 리소스에 액세스하도록 허용하는 정책을 업데이트하라고 관리자에게 요청합니다.

iam:PassRole을 수행하도록 인증되지 않음

`iam:PassRole` 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 MemoryDB에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스 에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 역할을 서비스에 전달할 권한이 있어야 합니다.

다음 예제 오류는 `marymajor`(이)라는 IAM 사용자가 콘솔을 사용하여 MemoryDB에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 권한이 없습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우, Mary가 `iam:PassRole`작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

내 AWS 계정 외부의 사람이 내 MemoryDB 리소스에 액세스하도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우, 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세한 내용은 다음을 참조하세요.

- MemoryDB에서 이러한 기능을 지원하는지 여부를 알아보려면 MemoryDB에서 IAM을 사용하는 방식 단원을 참조하세요.

- 소유 AWS 계정 한의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 소유한 다른의 IAM 사용자에게 액세스 권한 제공을 참조 AWS 계정 하세요.
- 타사에 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사가 AWS 계정 소유한에 대한 액세스 권한 제공을 AWS 계정참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용자 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하세요.
- 크로스 계정 액세스에 대한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용자 설명서의 IAM의 크로스 계정 리소스 액세스를 참조하세요.

액세스 관리

요청을 인증하는 데 유효한 자격 증명이 있더라도 권한이 없다면 MemoryDB 리소스를 생성하거나 액세스할 수 없습니다. 예를 들어, MemoryDB 클러스터를 생성할 권한이 있어야 합니다.

다음 섹션에서는 MemoryDB에 대한 권한을 관리하는 방법을 설명합니다. 먼저 개요를 읽어보면 도움이 됩니다.

- MemoryDB 리소스에 대한 액세스 권한 관리 개요
- MemoryDB에 대한 ID 기반 정책(IAM 정책) 사용

MemoryDB 리소스에 대한 액세스 권한 관리 개요

모든 AWS 리소스는 AWS 계정이 소유하며, 리소스를 생성하거나 액세스할 수 있는 권한은 권한 정책에 의해 관리됩니다. 계정 관리자는 IAM ID(사용자, 그룹 및 역할)에 권한 정책을 연결할 수 있습니다. 또한 MemoryDB에서는 리소스에 권한 정책 연결을 지원합니다.

> **Note**
>
> 계정 관리자 또는 관리자 사용자는 관리자 권한이 있는 사용자입니다. 자세한 설명은 IAM 사용자 가이드의 IAM 모범 사례 섹션을 참조하십시오.

액세스 권한을 제공하려면 사용자, 그룹 또는 역할에 권한을 추가하세요.

- 의 사용자 및 그룹 AWS IAM Identity Center:

  권한 세트를 생성합니다. AWS IAM Identity Center 사용자 안내서에서 권한 세트 생성의 지침을 따릅니다.
- ID 제공업체를 통해 IAM에서 관리되는 사용자:

  ID 페더레이션을 위한 역할을 생성합니다. IAM 사용자 설명서의 Create a role for a third-party identity provider (federation)의 지침을 따릅니다.
- IAM 사용자:
  - 사용자가 맡을 수 있는 역할을 생성합니다. IAM 사용자 설명서에서 Create a role for an IAM user의 지침을 따릅니다.
  - (권장되지 않음) 정책을 사용자에게 직접 연결하거나 사용자를 사용자 그룹에 추가합니다. IAM 사용 설명서에서 사용자(콘솔)에 권한 추가의 지침을 따릅니다.

주제

- AWS MemoryDB에 대한 관리형 정책
- MemoryDB API 권한: 작업, 리소스 및 조건 참조

MemoryDB 리소스 및 작업

MemoryDB의 기본 리소스는 클러스터입니다.

다음에 나와 있는 것처럼 이러한 리소스에는 고유한 Amazon 리소스 이름(ARN)이 연결됩니다.

> **Note**
> 리소스 수준 권한이 유효하려면 ARN 문자열의 리소스 이름이 소문자여야 합니다.

리소스 유형	ARN 형식
User	arn:aws:memorydb:*us-east-1:123456789012* :user/user1
액세스 제어 목록(ACL)	arn:aws:memorydb:*us-east-1:123456789012* :acl/myacl
Cluster	arn:aws:memorydb:*us-east-1:123456789012* :cluster/my-cluster
스냅샷	arn:aws:memorydb:*us-east-1:123456789012* :snapshot/my-snapshot
파라미터 그룹	arn:aws:memorydb:*us-east-1:123456789012* :parametergroup/my-parameter-group
서브넷 그룹	arn:aws:memorydb:*us-east-1:123456789012* :subnetgroup/my-subnet-group

MemoryDB는 MemoryDB 리소스를 처리하기 위한 일련의 작업을 제공합니다. 사용 가능한 작업 목록은 MemoryDB 작업을 참조하세요.

리소스 소유권 이해

리소스 소유자는 리소스를 생성한 AWS 계정입니다. 즉, 리소스 소유자는 리소스를 생성하는 요청을 인증하는 보안 주체 엔터티의 AWS 계정입니다. 보안주체 엔터티는 루트 계정, IAM 사용자 또는 IAM 역할일 수 있습니다. 다음 예제에서는 이러한 작동 방법을 설명합니다.

- AWS 계정의 루트 계정 자격 증명을 사용하여 클러스터를 생성한다고 가정해 보겠습니다. 이 경우 AWS 계정은 리소스의 소유자입니다. MemoryDB에서 리소스는 클러스터입니다.
- AWS 계정에서 IAM 사용자를 생성하고 해당 사용자에게 클러스터를 생성할 수 있는 권한을 부여한다고 가정해 보겠습니다. 이러한 경우, 이 사용자가 클러스터를 생성할 수 있습니다. 하지만 사용자가 속한 AWS 계정이 클러스터 리소스를 소유합니다.
- AWS 계정에서 클러스터를 생성할 권한이 있는 IAM 역할을 생성한다고 가정해 보겠습니다. 이러한 경우, 이 역할을 수임할 사람은 클러스터를 생성할 수 있습니다. 역할이 속한 AWS 계정이 클러스터 리소스를 소유합니다.

리소스 액세스 관리

권한 정책은 누가 무엇에 액세스 할 수 있는지를 나타냅니다. 다음 섹션에서는 권한 정책을 만드는 데 사용 가능한 옵션에 대해 설명합니다.

> **Note**
>
> 이 섹션에서는 MemoryDB의 맥락에서 IAM을 사용하는 방법을 설명하며, IAM 서비스에 대한 자세한 내용은 다루지 않습니다. IAM 설명서의 전체 내용은 IAM 사용자 안내서의 IAM이란?을 참조하세요. IAM 정책 구문과 설명에 대한 자세한 내용은 IAM 사용 설명서의 AWS IAM 정책 참조 섹션을 참조하세요.

IAM 보안 인증에 연결된 정책을 보안 인증 기반 정책(IAM 정책)이라고 합니다. 리소스에 연결된 정책을 리소스 기반 정책이라고 합니다.

주제

- 자격 증명 기반 정책(IAM 정책)
- 정책 요소 지정: 작업, 효과, 리소스, 보안 주체
- 정책에서 조건 지정

자격 증명 기반 정책(IAM 정책)

정책을 IAM 보안 인증에 연결할 수 있습니다. 예를 들면, 다음을 수행할 수 있습니다.

- 계정 내 사용자 또는 그룹에 권한 정책 연결 - 계정 관리자는 권한을 부여하기 위해 특정 사용자에 연결된 권한 정책을 사용할 수 있습니다. 이 경우, 해당 사용자는 클러스터, 파라미터 그룹 또는 보안 그룹과 같은 MemoryDB 리소스를 생성할 수 있습니다.
- 역할에 권한 정책 연결(교차 계정 권한 부여) - 자격 증명 기반 권한 정책을 IAM 역할에 연결하여 교차 계정 권한을 부여할 수 있습니다. 예를 들어 계정 A의 관리자는 다음과 같이 다른 AWS 계정(예: 계정 B) 또는 AWS 서비스에 교차 계정 권한을 부여하는 역할을 생성할 수 있습니다.
  1. 계정 A 관리자는 IAM 역할을 생성하고 계정 A의 리소스에 대한 권한을 부여하는 역할에 권한 정책을 연결합니다.
  2. 계정 A 관리자는 계정 B를 역할을 수임할 보안 주체로 식별하는 역할에 신뢰 정책을 연결합니다.
  3. 그런 다음 계정 B 관리자는 계정 B의 사용자에게 역할을 수임할 수 있는 권한을 위임할 수 있습니다. 이렇게 하면 계정 B의 사용자가 계정 A의 리소스를 생성하거나 액세스할 수 있습니다. 경우에 따라 역할을 수임할 수 있는 AWS 서비스 권한을 부여할 수 있습니다. 이러한 접근 방식을 지원하려면, 신뢰 정책의 보안 주체는 AWS 서비스 보안 주체일 수도 있습니다.

  IAM을 사용하여 권한을 위임하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 액세스 관리 섹션을 참조하세요.

다음은 사용자가 AWS 계정에 대해 `DescribeClusters` 작업을 수행하도록 허용하는 정책의 예입니다. 또한 MemoryDB는 API 작업에 대한 리소스 ARN을 사용하여 특정 리소스를 식별할 수 있도록 지원합니다. (이러한 접근 방식을 리소스 수준 권한이라고도 합니다.)

MemoryDB에서 보안 인증 기반 정책을 사용하는 방법에 대한 자세한 내용은 MemoryDB에 대한 ID 기반 정책(IAM 정책) 사용 단원을 참조하세요. 사용자, 그룹, 역할 및 권한에 대한 자세한 내용은 IAM 사용 설명서의 자격 증명(사용자, 그룹 및 역할)을 참조하세요.

정책 요소 지정: 작업, 효과, 리소스, 보안 주체

각 MemoryDB 리소스(MemoryDB 리소스 및 작업 참조)에 대해 서비스는 일련의 API 작업(작업 참조)을 정의합니다. 이러한 API 작업에 대한 권한을 부여하기 위해 MemoryDB에서는 정책에서 지정할 수 있는 작업을 정의합니다. 예를 들어 MemoryDB 클러스터 리소스에 대해 `CreateCluster`, `DeleteCluster` 및 `DescribeClusters` 작업을 정의합니다. API 작업을 실시하려면 둘 이상의 작업에 대한 권한이 필요할 수 있습니다.

다음은 가장 기본적인 정책 요소입니다.

- 리소스 – 정책에서 Amazon 리소스 이름(ARN)을 사용하여 정책을 적용할 리소스를 식별합니다. 자세한 내용은 MemoryDB 리소스 및 작업 단원을 참조하십시오.
- 작업 – 작업 키워드를 사용하여 허용 또는 거부할 리소스 작업을 식별합니다. 예를 들면, 지정된 `Effect`에 따라 `memorydb:CreateCluster` 권한은 MemoryDB `CreateCluster` 작업을 수행할 수 있는 사용자 권한을 허용하거나 거부합니다.
- 결과 – 사용자가 특정 작업을 요청하는 경우의 결과를 지정합니다. 이는 허용 또는 거부 중에 하나가 될 수 있습니다. 명시적으로 리소스에 대한 액세스 권한을 부여(허용)하지 않는 경우, 액세스는 묵시적으로 거부됩니다. 리소스에 대한 액세스를 명시적으로 거부할 수도 있습니다. 예를 들어, 다른 정책에서 액세스 권한을 부여하더라도 사용자가 해당 리소스에 액세스할 수 없도록 하려고 할 때 이러한 작업을 수행할 수 있습니다.
- 보안 주체 – 자격 증명 기반 정책(IAM 정책)에서 정책이 연결되는 사용자는 암시적인 보안 주체입니다. 리소스 기반 정책의 경우, 사용자, 계정, 서비스 또는 권한의 수신자인 기타 개체를 지정합니다(리소스 기반 정책에만 해당).

IAM 정책 구문과 설명에 대한 자세한 내용은 IAM 사용 설명서의 AWS IAM 정책 참조 섹션을 참조하세요.

모든 MemoryDB API 작업을 보여 주는 표는 MemoryDB API 권한: 작업, 리소스 및 조건 참조 섹션을 참조하세요.

정책에서 조건 지정

권한을 부여할 때 IAM 정책 언어를 사용하여 정책이 적용되는 조건을 지정할 수 있습니다. 예를 들어, 특정 날짜 이후에만 정책을 적용할 수 있습니다. 정책 언어에서의 조건 지정에 관한 자세한 내용은 IAM 사용자 안내서의 조건을 참조하세요.

MemoryDB에 대한 ID 기반 정책(IAM 정책) 사용

이 항목에서는 계정 관리자가 IAM 자격 증명(사용자, 그룹, 역할)에 권한 정책을 연결할 수 있는 자격 증명 기반 정책의 예를 제공합니다.

> **Important**
>
> MemoryDB 리소스 액세스를 관리하기 위한 기본 개념과 옵션을 설명하는 주제를 먼저 읽어보는 것이 좋습니다. 자세한 내용은 MemoryDB 리소스에 대한 액세스 권한 관리 개요 단원을 참조하십시오.

이 주제의 섹션에서는 다음 내용을 학습합니다.

- MemoryDB 콘솔 사용에 필요한 권한
- MemoryDB에 대한AWS관리형 (사전 정의된) 정책
- 고객 관리형 정책 예제

다음은 권한 정책의 예입니다.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [{
       "Sid": "AllowClusterPermissions",
       "Effect": "Allow",
       "Action": [
          "memorydb:CreateCluster",
          "memorydb:DescribeClusters",
          "memorydb:UpdateCluster"],
       "Resource": "*"
       },
       {
         "Sid": "AllowUserToPassRole",
         "Effect": "Allow",
         "Action": [ "iam:PassRole" ],
         "Resource": "arn:aws:iam::123456789012:role/EC2-roles-for-cluster"
       }
```

```
    ]
}
```

이 정책에는 두 명령문이 있습니다:

- 첫 번째 명령문은 계정이 소유하는 임의의 클러스터에서 MemoryDB 작업(`memorydb:CreateCluster`, `memorydb:DescribeClusters`, 및 `memorydb:UpdateCluster`)에 대한 권한을 부여합니다.
- 두 번째 명령문은 `Resource` 값의 끝에 지정된 IAM 역할 이름에서 IAM 작업(`iam:PassRole`)에 대한 권한을 부여합니다.

자격 증명 기반 정책에서는 권한을 가질 보안 주체를 지정하지 않으므로 이 정책은 `Principal` 요소를 지정하지 않습니다. 정책을 사용자에게 연결할 경우, 사용자는 암시적인 보안 주체입니다. IAM 역할에 권한 정책을 연결하면 역할의 신뢰 정책에서 식별된 보안 주체가 권한을 얻습니다.

모든 MemoryDB API 작업과 해당 작업이 적용되는 리소스를 보여주는 표는 MemoryDB API 권한: 작업, 리소스 및 조건 참조 섹션을 참조하세요.

MemoryDB 콘솔 사용에 필요한 권한

권한 참조 표에 MemoryDB API 작업과 각 작업에 필요한 권한이 나열되어 있습니다. MemoryDB API 작업에 대한 자세한 내용은 MemoryDB API 권한: 작업, 리소스 및 조건 참조 단원을 참조하세요.

MemoryDB 콘솔을 사용하려면 먼저 다음 권한 정책과 같이 추가 작업에 대한 권한을 부여해야 합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "MinPermsForMemDBConsole",
        "Effect": "Allow",
        "Action": [
            "memorydb:Describe*",
            "memorydb:List*",
            "ec2:DescribeAvailabilityZones",
            "ec2:DescribeVpcs",
            "ec2:DescribeAccountAttributes",
            "ec2:DescribeSecurityGroups",
```

```
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:DescribeAlarms",
                "s3:ListAllMyBuckets",
                "sns:ListTopics",
                "sns:ListSubscriptions" ],
            "Resource": "*"
            }
        ]
}
```

MemoryDB 콘솔에 이러한 추가 권한이 필요한 이유는 다음과 같습니다.

- MemoryDB 작업 권한이 있으면 콘솔에서 계정의 MemoryDB 리소스를 표시할 수 있습니다.
- 콘솔에 Amazon EC2 쿼리를 위한 `ec2` 작업 권한이 있어야 합니다. 그래야 가용 영역, VPC, 보안 그룹 및 계정 속성을 표시할 수 있습니다.
- `cloudwatch` 작업에 대한 권한이 있으면 콘솔에서 Amazon CloudWatch 지표와 경보를 검색하고 콘솔에 표시할 수 있습니다.
- `sns` 작업에 대한 권한이 있으면 콘솔에서 Amazon Simple Notification Service(Amazon SNS) 주제와 구독을 검색하고 콘솔에 표시할 수 있습니다.

고객 관리형 정책 예제

기본 정책을 사용하지 않고 고객 관리형 정책을 사용할 경우, 두 가지 중 하나가 가능한지 확인해야 합니다. `iam:createServiceLinkedRole`을 호출할 수 있는 권한이 있어야 합니다(자세한 내용은 예제 4: 사용자에게 IAM CreateServiceLinkedRole API 호출 허용 섹션을 참조하세요). 또는 MemoryDB 서비스 연결 역할을 생성해야 합니다.

MemoryDB 콘솔을 사용하는 데 필요한 최소 권한과 함께 이 섹션의 예제 정책을 사용할 경우 추가 권한이 부여됩니다. 이 예제는 AWS SDKs 및 와도 관련이 있습니다 AWS CLI. MemoryDB 콘솔을 사용하는 데 필요한 권한에 대한 자세한 내용은 MemoryDB 콘솔 사용에 필요한 권한 단원을 참조하세요.

IAM 사용자 및 그룹을 설정하는 것에 대한 지침은 IAM 사용 설명서의 IAM 사용자와 관리자 그룹 처음 만들기 섹션을 참조하세요.

> **Important**
>
> 프로덕션에서 IAM 정책을 사용하기 전에 항상 철저히 테스트하세요. 나타나는 일부 MemoryDB 작업을 MemoryDB 콘솔을 사용할 때 다른 작업이 해당 작업을 지원해야 할 수도

있습니다. 예를 들어, `memorydb:CreateCluster`이(가) MemoryDB 클러스터를 생성하기 위한 권한을 부여합니다. 그러나 이 작업을 수행하기 위해 MemoryDB 콘솔에서 여러 `Describe` 및 `List` 작업을 사용하여 콘솔 목록을 채웁니다.

예제

- 예제 1: 사용자에게 MemoryDB 리소스에 대한 읽기 전용 액세스 허용
- 예제 2: 사용자가 일반 MemoryDB 시스템 관리자 작업을 수행하도록 허용
- 예제 3: 사용자가 모든 MemoryDB API 작업에 액세스하도록 허용
- 예제 4: 사용자에게 IAM CreateServiceLinkedRole API 호출 허용

예제 1: 사용자에게 MemoryDB 리소스에 대한 읽기 전용 액세스 허용

다음 정책은 사용자에게 리소스를 나열하도록 허용하는 MemoryDB 작업에 대한 권한을 부여합니다. 일반적으로 이 유형의 권한 정책을 관리자 그룹에 연결합니다.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[{
      "Sid": "MemDBUnrestricted",
      "Effect":"Allow",
      "Action": [
          "memorydb:Describe*",
          "memorydb:List*"],
      "Resource":"*"
      }
   ]
}
```

예제 2: 사용자가 일반 MemoryDB 시스템 관리자 작업을 수행하도록 허용

일반 시스템 관리자 작업에는 클러스터, 파라미터 및 파라미터 그룹 수정이 포함됩니다. 시스템 관리자가 MemoryDB 이벤트에 대한 정보를 보고 싶어할 수도 있습니다. 다음 정책은 이러한 일반 시스템 관리자 작업을 위한 MemoryDB 작업을 수행할 권한을 사용자에게 부여합니다. 일반적으로 이 유형의 권한 정책을 시스템 관리자 그룹에 연결합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "MDBAllowSpecific",
            "Effect": "Allow",
            "Action": [
                "memorydb:UpdateCluster",
                "memorydb:DescribeClusters",
                "memorydb:DescribeEvents",
                "memorydb:UpdateParameterGroup",
                "memorydb:DescribeParameterGroups",
                "memorydb:DescribeParameters",
                "memorydb:ResetParameterGroup"
            ],
            "Resource": "*"
        }
    ]
}
```

예제 3: 사용자가 모든 MemoryDB API 작업에 액세스하도록 허용

다음 정책은 사용자가 모든 MemoryDB 작업을 호출할 수 있도록 허용합니다. 관리자 사용자에게만 이 유형의 권한 정책을 부여하는 것이 좋습니다.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[{
      "Sid": "MDBAllowAll",
      "Effect":"Allow",
      "Action":[
          "memorydb:*" ],
      "Resource":"*"
      }
   ]
}
```

예제 4: 사용자에게 IAM CreateServiceLinkedRole API 호출 허용

다음 정책은 사용자가 IAM `CreateServiceLinkedRole` API를 호출하도록 허용합니다. 변화하기 쉬운 MemoryDB 작업을 간접적으로 호출하는 사용자에게 이 유형의 권한 정책을 부여하는 것이 좋습니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"CreateSLRAllows",
      "Effect":"Allow",
      "Action":[
        "iam:CreateServiceLinkedRole"
      ],
      "Resource":"*",
      "Condition":{
        "StringLike":{
          "iam:AWS ServiceName":"memorydb.amazonaws.com"
        }
      }
    }
  ]
}
```

리소스 수준 권한

IAM 정책에 리소스를 지정하여 권한의 범위를 제한할 수 있습니다. 많은 AWS CLI API 작업은 작업의 동작에 따라 달라지는 리소스 유형을 지원합니다. 각 IAM 정책 구문은 리소스에 대해 수행되는 작업에 대한 권한을 부여합니다. 지명된 리소스에서 이루어지는 작업이 아니거나 모든 리소스에 대해 그 작업을 수행할 수 있도록 권한을 부여하는 경우, 정책에서 해당 리소스의 값은 와일드카드(*)가 됩니다. 대부분의 API 작업에서는 리소스의 Amazon 리소스 이름(ARN) 또는 복수의 리소스에 맞는 ARN 패턴을 지정함으로써 사용자 수정이 가능한 리소스를 제한할 수 있습니다. 리소스별 권한을 제한하려면 ARN으로 리소스를 지정하세요.

MemoryDB 리소스 ARN 포맷

> **Note**
>
> 리소스 수준 권한이 유효하려면 ARN 문자열의 리소스 이름이 소문자여야 합니다.

- *### - arn:aws:memorydb:us-east-1:123456789012*:user/user1
- ACL – arn:aws:memorydb:*us-east-1:123456789012*:acl/my-acl
- 클러스터 – arn:aws:memorydb:*us-east-1:123456789012*:cluster/my-cluster
- 스냅샷 – arn:aws:memorydb:*us-east-1:123456789012*:snapshot/my-snapshot
- *#### ## - arn:aws:memorydb:us-east-1:123456789012*:parametergroup/my-parameter-group
- *### ## - arn:aws:memorydb:us-east-1:123456789012*:subnetgroup/my-subnet-group

예제

- 예제 1: 사용자에게 특정 MemoryDB 리소스 유형에 대한 모든 액세스 권한 허용
- 예 2: 클러스터에 대한 사용자 액세스 거부

예제 1: 사용자에게 특정 MemoryDB 리소스 유형에 대한 모든 액세스 권한 허용

다음 정책은 서브넷 그룹, 보안 그룹 및 클러스터 유형의 모든 리소스에 대해 지정된 `account-id` 전체 액세스를 명시적으로 허용합니다.

```
{
        "Sid": "Example1",
        "Effect": "Allow",
        "Action": "memorydb:*",
        "Resource": [
             "arn:aws:memorydb:us-east-1:account-id:subnetgroup/*",
             "arn:aws:memorydb:us-east-1:account-id:securitygroup/*",
             "arn:aws:memorydb:us-east-1:account-id:cluster/*"
        ]
}
```

예 2: 클러스터에 대한 사용자 액세스 거부

다음 예제에서는 특정 클러스터에 대한 지정된 `account-id` 액세스를 명시적으로 거부합니다.

```
{
        "Sid": "Example2",
        "Effect": "Deny",
        "Action": "memorydb:*",
        "Resource": [
                "arn:aws:memorydb:us-east-1:account-id:cluster/name"
        ]
}
```

MemoryDB에 대한 서비스 연결 역할 사용

MemoryDB는 AWS Identity and Access Management (IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 MemoryDB와 같은 AWS 서비스에 직접 연결된 고유한 유형의 IAM 역할입니다. MemoryDB 서비스 연결 역할은 MemoryDB에 의해 사전 정의됩니다. 서비스에서 클러스터를 대신하여 AWS 서비스를 호출하기 위해 필요한 모든 권한을 포함합니다.

서비스 연결 역할을 사용하면 필요한 권한을 수동으로 추가할 필요가 없으므로 MemoryDB를 더 쉽게 설정할 수 있습니다. 역할은 AWS 계정 내에 이미 존재하지만 MemoryDB 사용 사례에 연결되고 사전 정의된 권한이 있습니다. MemoryDB만 이러한 역할을 맡을 수 있고 이러한 역할만 사전 정의된 권한 정책을 사용할 수 있습니다. 먼저 역할의 관련 리소스를 삭제해야만 역할을 삭제할 수 있습니다. 이렇게 하면 MemoryDB 리소스에 대한 필수 액세스 권한을 부주의로 삭제할 수 없기 때문에 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용은 IAM으로 작업하는AWS 서비스를 참조해 서비스 연결 역할 열이 예(Yes)인 서비스를 찾으세요. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예(Yes) 링크를 선택합니다.

목차

MemoryDB에 대한 서비스 연결 역할 권한

MemoryDB는 AWSServiceRoleForMemoryDB라는 서비스 연결 역할을 사용합니다.이 정책은 MemoryDB가 클러스터 관리에 필요한 대로 사용자를 대신하여 AWS 리소스를 관리할 수 있도록 허용합니다.

AWSServiceRoleForMemoryDB 서비스 연결 역할의 권한 정책은 MemoryDB가 지정된 리소스에 대해 다음 작업을 수행하도록 허용합니다.

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "ec2:CreateTags"
   ],
   "Resource": "arn:aws-cn:ec2:*:*:network-interface/*",
   "Condition": {
    "StringEquals": {
     "ec2:CreateAction": "CreateNetworkInterface"
    },
    "ForAllValues:StringEquals": {
     "aws:TagKeys": [
      "AmazonMemoryDBManaged"
     ]
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
```

```
    "ec2:CreateNetworkInterface"
   ],
   "Resource": [
    "arn:aws-cn:ec2:*:*:network-interface/*",
    "arn:aws-cn:ec2:*:*:subnet/*",
    "arn:aws-cn:ec2:*:*:security-group/*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "ec2:DeleteNetworkInterface",
    "ec2:ModifyNetworkInterfaceAttribute"
   ],
   "Resource": "arn:aws-cn:ec2:*:*:network-interface/*",
   "Condition": {
    "StringEquals": {
     "ec2:ResourceTag/AmazonMemoryDBManaged": "true"
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
    "ec2:DeleteNetworkInterface",
    "ec2:ModifyNetworkInterfaceAttribute"
   ],
   "Resource": "arn:aws-cn:ec2:*:*:security-group/*"
  },
  {
   "Effect": "Allow",
   "Action": [
    "ec2:DescribeSecurityGroups",
    "ec2:DescribeNetworkInterfaces",
    "ec2:DescribeAvailabilityZones",
    "ec2:DescribeSubnets",
    "ec2:DescribeVpcs"
   ],
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": [
    "cloudwatch:PutMetricData"
```

```
      ],
      "Resource": "*",
      "Condition": {
       "StringEquals": {
        "cloudwatch:namespace": "AWS/MemoryDB"
       }
      }
     }
    ]
   }
```

자세한 내용은 AWS 관리형 정책: MemoryDBServiceRolePolicy 단원을 참조하십시오.

IAM 엔터티가 AWSServiceRoleForMemoryDB 서비스 연결 역할을 생성하도록 허용

IAM 개체에 대한 권한에 다음 정책 설명을 추가합니다.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:CreateServiceLinkedRole",
        "iam:PutRolePolicy"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/memorydb.amazonaws.com/
AWSServiceRoleForMemoryDB*",
    "Condition": {"StringLike": {"iam:AWS ServiceName": "memorydb.amazonaws.com"}}
}
```

IAM 엔터티가 AWSServiceRoleForMemoryDB 서비스 연결 역할을 삭제하도록 허용

IAM 개체에 대한 권한에 다음 정책 설명을 추가합니다.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:DeleteServiceLinkedRole",
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/memorydb.amazonaws.com/
AWSServiceRoleForMemoryDB*",
    "Condition": {"StringLike": {"iam:AWS ServiceName": "memorydb.amazonaws.com"}}
}
```

또는 AWS 관리형 정책을 사용하여 MemoryDB에 대한 전체 액세스를 제공할 수 있습니다.

서비스 연결 역할 생성(IAM)

IAM 콘솔, CLI 또는 API를 사용하여 서비스 연결 역할을 생성할 수 있습니다.

서비스 연결 역할 생성(IAM 콘솔)

IAM 콘솔을 사용하여 서비스 연결 역할을 생성할 수 있습니다.

서비스 연결 역할을 만드는 방법(콘솔)

1. 에 로그인 AWS Management 콘솔 하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.
2. IAM 콘솔의 왼쪽 탐색 창에서 역할(Roles)을 선택합니다. 그런 다음 새 역할 생성(Create new role)을 선택합니다.
3. 신뢰할 수 있는 유형의 엔터티 선택 아래에서 AWS 서비스를 선택합니다.
4. 또는 사용 사례를 표시하도록 서비스 선택(Or select a service to view its use cases)에서 MemoryDB를 선택합니다.
5. 다음: 권한을 선택합니다.
6. 정책 이름에서 이 역할에 `MemoryDBServiceRolePolicy`가 있는지 확인합니다. 다음: 태그를 선택합니다.
7. 서비스 연결 역할에는 태그가 지원되지 않습니다. 다음: 검토를 선택합니다.
8. (선택 사항) 역할 설명(Role description)에서 새로운 서비스 연결 역할에 대한 설명을 편집합니다.
9. 역할을 검토한 다음 역할 생성을 선택합니다.

서비스 연결 역할 생성(IAM CLI)

에서 IAM 작업을 사용하여 서비스 연결 역할을 AWS Command Line Interface 생성할 수 있습니다. 이 역할에는 서비스가 역할을 수임하는 데 필요한 신뢰 정책 및 인라인 정책이 포함될 수 있습니다.

서비스 연결 역할을 만드는 방법(CLI)

다음 작업을 사용합니다.

```
$ aws iam create-service-linked-role --aws-service-name memorydb.amazonaws.com
```

서비스 연결 역할 생성(IAM API)

IAM API를 사용하여 서비스 연결 역할을 생성할 수 있습니다. 이 역할에는 서비스가 역할을 수임하는 데 필요한 신뢰 정책 및 인라인 정책이 포함될 수 있습니다.

서비스 연결 역할을 만드는 방법(API 사용)

CreateServiceLinkedRole API 호출을 사용합니다. 요청 시 `memorydb.amazonaws.com` 서비스 이름을 지정합니다.

MemoryDB에 대한 서비스 연결 역할의 설명 편집

MemoryDB는 AWSServiceRoleForMemoryDB 서비스 연결 역할을 편집하도록 허용하지 않습니다. 서비스 연결 역할을 생성한 후에는 다양한 개체가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. 하지만 IAM을 사용하여 역할의 설명을 편집할 수 있습니다.

서비스 연결 역할 설명 편집(IAM 콘솔)

IAM 콘솔을 사용하여 서비스 연결 역할의 설명을 편집할 수 있습니다.

서비스 연결 역할의 설명을 편집하는 방법(콘솔)

1. IAM 콘솔의 왼쪽 탐색 창에서 역할(Roles)을 선택합니다.
2. 변경할 역할 이름을 선택합니다.
3. 역할 설명의 맨 오른쪽에서 편집을 선택합니다.
4. 상자에 새 설명을 입력하고 저장을 선택합니다.

서비스 연결 역할 설명 편집(IAM CLI)

에서 IAM 작업을 사용하여 서비스 연결 역할 설명을 AWS Command Line Interface 편집할 수 있습니다.

서비스 연결 역할의 설명을 변경하는 방법(CLI)

1. (선택 사항) 역할에 대한 현재 설명을 보려면 IAM 작업 AWS CLI 용를 사용합니다`get-role`.

   Example

   ```
   $ aws iam get-role --role-name AWSServiceRoleForMemoryDB
   ```

CLI 작업에서 역할을 참조하려면 ARN이 아니라 역할 이름을 사용해야 합니다. 예를 들어 어떤 역할의 ARN이 `arn:aws:iam::123456789012:role/myrole`인 경우 참조할 역할은 **`myrole`**입니다.

2. 서비스 연결 역할의 설명을 업데이트하려면 IAM 작업 AWS CLI 용를 사용합니다[`update-role-description`](https://docs.aws.amazon.com/cli/latest/reference/iam/update-role-description.html).

   Linux, macOS, Unix의 경우:

   ```
   $ aws iam update-role-description \
       --role-name AWSServiceRoleForMemoryDB \
       --description "new description"
   ```

   Windows의 경우:

   ```
   $ aws iam update-role-description ^
       --role-name AWSServiceRoleForMemoryDB ^
       --description "new description"
   ```

서비스 연결 역할 설명 편집(IAM API)

IAM API를 사용하여 서비스 연결 역할의 설명을 편집할 수 있습니다.

서비스 연결 역할의 설명을 변경하는 방법(API)

1. (선택 사항) 역할의 현재 설명을 보려면 IAM API 작업 GetRole을 사용하세요.

   Example

   ```
   https://iam.amazonaws.com/
      ?Action=GetRole
      &RoleName=AWSServiceRoleForMemoryDB
      &Version=2010-05-08
      &AUTHPARAMS
   ```

2. 역할 설명을 업데이트하려면 IAM API 작업 UpdateRoleDescription을 사용하세요.

   Example

   ```
   https://iam.amazonaws.com/
   ```

```
?Action=UpdateRoleDescription
&RoleName=AWSServiceRoleForMemoryDB
&Version=2010-05-08
&Description="New description"
```

MemoryDB에 대한 서비스 연결 역할 삭제

서비스 연결 역할이 필요한 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제하는 것이 좋습니다. 따라서 적극적으로 모니터링하거나 유지하지 않는 미사용 엔터티가 없도록 합니다. 단, 삭제 전에 서비스 연결 역할을 정리해야 합니다.

MemoryDB에서는 서비스 연결 역할을 자동으로 삭제하지 않습니다.

서비스 연결 역할 정리

IAM을 사용하여 서비스 연결 역할을 삭제하기 전에 먼저 역할에 해당 역할과 연결된 리소스(클러스터)가 없는지 확인합니다.

IAM 콘솔에서 서비스 연결 역할에 활성 세션이 있는지 확인하려면

1. 에 로그인 AWS Management 콘솔 하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.
2. IAM 콘솔의 왼쪽 탐색 창에서 역할(Roles)을 선택합니다. 그런 다음 AWSServiceRoleForMemoryDB 역할의 이름(확인란 아님)을 선택합니다.
3. 선택한 역할의 요약 페이지에서 Access Advisor 탭을 선택합니다.
4. Access Advisor 탭에서 서비스 연결 역할의 최근 활동을 검토합니다.

AWSServiceRoleForMemoryDB가 필요한 MemoryDB 리소스를 삭제하려면(콘솔)

- 클러스터를 삭제하려면 다음을 참조하세요.
  - 사용AWS Management 콘솔
  - 사용AWS CLI
  - MemoryDB API 사용

서비스 연결 역할 삭제(IAM 콘솔)

IAM 콘솔을 사용하여 서비스 연결 역할을 삭제할 수 있습니다.

서비스 연결 역할을 삭제하는 방법(콘솔)

1. 에 로그인 AWS Management 콘솔 하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.
2. IAM 콘솔의 왼쪽 탐색 창에서 역할(Roles)을 선택합니다. 그런 다음 삭제할 역할의 이름이나 행이 아닌 이름 옆에 있는 확인란을 선택합니다.
3. 페이지 상단의 역할 작업에서 역할 삭제를 선택합니다.
4. 확인 페이지에서 서비스에서 마지막으로 액세스한 데이터를 검토합니다. 그러면 선택한 각 역할이 AWS 서비스에 마지막으로 액세스한 시기가 표시됩니다. 이를 통해 역할이 현재 활동 중인지를 확인할 수 있습니다. 계속 진행하려면 예, 삭제합니다를 선택하여 삭제할 서비스 연결 역할을 제출합니다.
5. IAM 콘솔 알림을 보고 서비스 연결 역할 삭제 진행 상황을 모니터링합니다. IAM 서비스 연결 역할 삭제는 비동기이므로 삭제할 역할을 제출한 후에 삭제 태스크가 성공하거나 실패할 수 있습니다. 태스크에 실패할 경우 알림의 세부 정보 보기 또는 리소스 보기를 선택하면 삭제 실패 이유를 확인할 수 있습니다.

서비스 연결 역할 삭제(IAM CLI)

에서 IAM 작업을 사용하여 서비스 연결 역할을 AWS Command Line Interface 삭제할 수 있습니다.

서비스 연결 역할을 삭제하는 방법(CLI)

1. 삭제할 서비스 연결 역할의 이름을 모르는 경우 다음 명령을 입력합니다. 이 명령은 계정의 역할과 해당 Amazon 리소스 이름(ARN)을 나열합니다.

   ```
   $ aws iam get-role --role-name role-name
   ```

   CLI 작업에서 역할을 참조하려면 ARN이 아니라 역할 이름을 사용해야 합니다. 예를 들어 역할의 ARN이 `arn:aws:iam::123456789012:role/myrole`인 경우 해당 역할을 **myrole**으로 참조합니다.

2. 서비스 연결 역할이 사용되지 않거나 연결된 리소스가 없는 경우에는 서비스 연결 역할을 삭제할 수 없으므로 삭제 요청을 제출해야 합니다. 이러한 조건이 충족되지 않으면 요청이 거부될 수 있습니다. 삭제 태스크 상태를 확인하려면 응답의 `deletion-task-id`(을)를 캡처해야 합니다. 다음을 입력하여 서비스 연결 역할 삭제 요청을 제출합니다.

   ```
   $ aws iam delete-service-linked-role --role-name role-name
   ```

3. 다음을 입력하여 삭제 작업의 상태를 확인합니다.

```
$ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
```

삭제 태스크는 NOT_STARTED, IN_PROGRESS, SUCCEEDED 또는 FAILED 상태일 수 있습니다. 삭제에 실패할 경우 문제를 해결할 수 있도록 실패 이유가 호출에 반환됩니다.

서비스 연결 역할 삭제(IAM API)

IAM API를 사용하여 서비스 연결 역할을 삭제할 수 있습니다.

서비스 연결 역할(API)을 삭제하는 방법

1. 서비스 연결 역할 삭제 요청을 제출하려면 DeleteServiceLinkedRole을 호출합니다. 요청에 역할 이름을 지정합니다.

   서비스 연결 역할이 사용되지 않거나 연결된 리소스가 없는 경우에는 서비스 연결 역할을 삭제할 수 없으므로 삭제 요청을 제출해야 합니다. 이러한 조건이 충족되지 않으면 요청이 거부될 수 있습니다. 삭제 태스크 상태를 확인하려면 응답의 DeletionTaskId(을)를 캡처해야 합니다.
2. 삭제 상태를 확인하려면 GetServiceLinkedRoleDeletionStatus을 호출합니다. 요청에 DeletionTaskId(을)를 지정합니다.

   삭제 태스크는 NOT_STARTED, IN_PROGRESS, SUCCEEDED 또는 FAILED 상태일 수 있습니다. 삭제에 실패할 경우 문제를 해결할 수 있도록 실패 이유가 호출에 반환됩니다.

AWS MemoryDB에 대한 관리형 정책

사용자, 그룹 및 역할에 권한을 추가하려면 직접 정책을 작성하는 것보다 AWS 관리형 정책을 사용하는 것이 더 쉽습니다. 팀에 필요한 권한만 제공하는 IAM 고객 관리형 정책을 생성하기 위해서는 시간과 전문 지식이 필요합니다. 빠르게 시작하려면 AWS 관리형 정책을 사용할 수 있습니다. 이러한 정책은 일반적인 사용 사례를 다루며 AWS 계정에서 사용할 수 있습니다. AWS 관리형 정책에 대한 자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책을 참조하세요.

AWS 서비스는 AWS 관리형 정책을 유지 관리하고 업데이트합니다. AWS 관리형 정책에서는 권한을 변경할 수 없습니다. 서비스에서 때때로 추가 권한을 AWS 관리형 정책에 추가하여 새로운 기능을 지원합니다. 이 유형의 업데이트는 정책이 연결된 모든 ID(사용자, 그룹 및 역할)에 적용됩니다. 서비스는

새로운 기능이 시작되거나 새 작업을 사용할 수 있을 때 AWS 관리형 정책에 업데이트됩니다. 서비스는 AWS 관리형 정책에서 권한을 제거하지 않으므로 정책 업데이트로 인해 기존 권한이 손상되지 않습니다.

또한 여러 서비스에 걸쳐 있는 직무에 대한 관리형 정책을 AWS 지원합니다. 예를 들어 ReadOnlyAccess AWS 관리형 정책은 모든 AWS 서비스 및 리소스에 대한 읽기 전용 액세스를 제공합니다. 서비스가 새 기능을 시작하면는 새 작업 및 리소스에 대한 읽기 전용 권한을 AWS 추가합니다. 직무 정책의 목록과 설명은 IAM 사용 설명서의 직무에 관한AWS 관리형 정책을 참조하세요.

AWS 관리형 정책: MemoryDBServiceRolePolicy

MemoryDBServiceRolePolicy AWS 관리형 정책을 계정의 자격 증명에 연결할 수 없습니다. 이 정책은 AWS MemoryDB 서비스 연결 역할의 일부입니다. 이 역할을 통해 서비스는 계정의 네트워크 인터페이스와 보안 그룹을 관리할 수 있습니다.

MemoryDB는 이 정책의 권한을 사용하여 EC2 보안 그룹 및 네트워크 인터페이스를 관리합니다. 이는 MemoryDB 클러스터를 관리하는 데 필요합니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "ec2:CreateTags"
   ],
   "Resource": "arn:aws-cn:ec2:*:*:network-interface/*",
   "Condition": {
    "StringEquals": {
     "ec2:CreateAction": "CreateNetworkInterface"
```

```
      },
      "ForAllValues:StringEquals": {
       "aws:TagKeys": [
        "AmazonMemoryDBManaged"
       ]
      }
     }
    },
    {
     "Effect": "Allow",
     "Action": [
      "ec2:CreateNetworkInterface"
     ],
     "Resource": [
      "arn:aws-cn:ec2:*:*:network-interface/*",
      "arn:aws-cn:ec2:*:*:subnet/*",
      "arn:aws-cn:ec2:*:*:security-group/*"
     ]
    },
    {
     "Effect": "Allow",
     "Action": [
      "ec2:DeleteNetworkInterface",
      "ec2:ModifyNetworkInterfaceAttribute"
     ],
     "Resource": "arn:aws-cn:ec2:*:*:network-interface/*",
     "Condition": {
      "StringEquals": {
       "ec2:ResourceTag/AmazonMemoryDBManaged": "true"
      }
     }
    },
    {
     "Effect": "Allow",
     "Action": [
      "ec2:DeleteNetworkInterface",
      "ec2:ModifyNetworkInterfaceAttribute"
     ],
     "Resource": "arn:aws-cn:ec2:*:*:security-group/*"
    },
    {
     "Effect": "Allow",
     "Action": [
      "ec2:DescribeSecurityGroups",
```

```
      "ec2:DescribeNetworkInterfaces",
      "ec2:DescribeAvailabilityZones",
      "ec2:DescribeSubnets",
      "ec2:DescribeVpcs"
     ],
     "Resource": "*"
    },
    {
     "Effect": "Allow",
     "Action": [
      "cloudwatch:PutMetricData"
     ],
     "Resource": "*",
     "Condition": {
      "StringEquals": {
       "cloudwatch:namespace": "AWS/MemoryDB"
      }
     }
    }
  ]
}
```

MemoryDB에 대한AWS관리형 (사전 정의된) 정책

AWS 는에서 생성하고 관리하는 독립 실행형 IAM 정책을 제공하여 많은 일반적인 사용 사례를 처리합니다 AWS. 관리형 정책은 필요한 권한을 사용자가 조사할 필요가 없도록 일반 사용 사례에 필요한 권한을 부여합니다. 자세한 내용은 IAM 사용자 안내서의 AWS 관리형 정책을 참조하세요.

계정의 사용자에게 연결할 수 있는 다음 AWS 관리형 정책은 MemoryDB에 고유합니다.

AmazonMemoryDBReadOnlyAccess

`AmazonMemoryDBReadOnlyAccess` 정책을 IAM ID에 연결할 수 있습니다. 이 정책은 모든 MemoryDB 리소스에 대한 읽기 전용 액세스를 허용하는 관리 권한을 부여합니다.

AmazonMemoryDBReadOnlyAccess - MemoryDB 리소스에 대한 읽기 전용 액세스 권한을 부여합니다.

JSON

```
{
```

```
 "Version":"2012-10-17",
 "Statement": [{
  "Effect": "Allow",
  "Action": [
   "memorydb:Describe*",
   "memorydb:List*"
  ],
  "Resource": "*"
 }]
}
```

AmazonMemoryDBFullAccess

`AmazonMemoryDBFullAccess` 정책을 IAM ID에 연결할 수 있습니다. 이 정책은 모든 MemoryDB 리소스에 대한 전체 액세스를 허용하는 관리 권한을 부여합니다.

AmazonMemoryDBFullAccess - 계정의 모든 MemoryDB 리소스에 대한 전체 액세스 권한을 부여합니다.

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [{
   "Effect": "Allow",
   "Action": "memorydb:*",
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": "iam:CreateServiceLinkedRole",
   "Resource": "arn:aws:iam::*:role/aws-service-role/memorydb.amazonaws.com/
AWSServiceRoleForMemoryDB",
   "Condition": {
    "StringLike": {
     "iam:AWSServiceName": "memorydb.amazonaws.com"
    }
   }
  }
 ]
}
```

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [{
   "Effect": "Allow",
   "Action": "memorydb:*",
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": "iam:CreateServiceLinkedRole",
   "Resource": "arn:aws-cn:iam::*:role/aws-service-role/memorydb.amazonaws.com/
AWSServiceRoleForMemoryDB",
   "Condition": {
    "StringLike": {
     "iam:AWSServiceName": "memorydb.amazonaws.com"
    }
   }
  }
 ]
}
```

고유의 사용자 지정 IAM 정책을 생성하여 MemoryDB API 작업에 대한 권한을 허용할 수도 있습니다. 해당 권한이 필요한 IAM 사용자 또는 그룹에 이러한 사용자 지정 정책을 연결할 수 있습니다.

AWS 관리형 정책에 대한 MemoryDB 업데이트

이 서비스가 이러한 변경 사항을 추적하기 시작한 이후부터 MemoryDB의 AWS 관리형 정책 업데이트에 대한 세부 정보를 봅니다. 이 페이지의 변경 사항에 대한 자동 알림을 받으려면 MemoryDB 문서 기록 페이지에서 RSS 피드를 구독하세요.

변경	설명	Date
AWS 관리형 정책: MemoryDBServiceRolePolicy – 정책 추가	MemoryDBServiceRolePolicy 가 memorydb:Replicate MultiRegionClusterData에 대	12/01/2024

변경	설명	Date
	한 권한을 추가했습니다. 이 권한을 통해 서비스 연결 역할은 MemoryDB Multi-Region 클러스터에 대한 데이터를 복제할 수 있습니다.	
AmazonMemoryDBFullAccess – 정책 추가	MemoryDB는 지원되는 리소스를 기술하고 나열하는 새 권한을 추가했습니다. 이러한 권한은 MemoryDB가 계정에서 지원되는 모든 리소스를 쿼리하는 데 필요합니다.	10/07/2021
AmazonMemoryDBReadOnlyAccess – 정책 추가	MemoryDB는 지원되는 리소스를 기술하고 나열하는 새 권한을 추가했습니다. 이러한 권한은 MemoryDB가 계정에서 지원되는 모든 리소스를 쿼리하여 계정 기반 애플리케이션을 생성하는 데 필요합니다.	10/07/2021
MemoryDB가 변경 사항 추적 시작	서비스 시작	8/19/2021

MemoryDB API 권한: 작업, 리소스 및 조건 참조

IAM 정책(보안 인증 기반 또는 리소스 기반)에 연결할 액세스 제어 및 쓰기 권한 정책을 설정할 때 다음 표를 참조로 사용하세요. 표에는 각 MemoryDB API 작업과 이 작업을 수행할 수 있는 권한을 부여할 수 있는 작업이 나와 있습니다. 정책의 `Action` 필드에서 작업을 지정하고, 정책의 `Resource` 필드에서 리소스 값을 지정합니다. 달리 명시되지 않는 한, 리소스는 필수입니다. 일부 필드에는 필수 리소스와 선택적 리소스가 모두 포함됩니다. 리소스 ARN이 없는 경우, 정책의 리소스는 와일드카드(*)입니다.

> **Note**
>
> 작업을 지정하려면 `memorydb:` 접두사 다음에 API 작업 명칭을 사용합니다(예: `memorydb:DescribeClusters`).

로깅 및 모니터링

MemoryDB 및 다른 AWS 솔루션의 신뢰성, 가용성 및 성능을 유지하려면 모니터링이 중요합니다. AWS는 MemoryDB를 모니터링하고, 이상이 있을 때 이를 보고하고, 필요한 경우, 자동 조치를 취할 수 있도록 다음과 같은 모니터링 도구를 제공합니다.

- Amazon CloudWatch는 AWS에서 실행하는 AWS 리소스와 애플리케이션을 실시간으로 모니터링합니다. 지표를 수집 및 추적하고, 사용자 지정 대시보드를 생성할 수 있으며, 지정된 지표가 지정한 임곗값에 도달하면 사용자에게 알리거나 조치를 취하도록 경보를 설정할 수 있습니다. 예를 들어 CloudWatch에서 Amazon EC2 인스턴스의 CPU 사용량 또는 기타 지표를 추적하고 필요할 때 자동으로 새 인스턴스를 시작할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하세요.
- Amazon CloudWatch Logs로 Amazon EC2 인스턴스, CloudTrail, 기타 소스의 로그 파일을 모니터링, 저장 및 액세스할 수 있습니다. CloudWatch Logs는 로그 파일의 정보를 모니터링하고 특정 임곗값에 도달하면 사용자에게 알릴 수 있습니다. 또한 매우 내구성이 뛰어난 스토리지에 로그 데이터를 저장할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서를 참조하세요.
- AWS CloudTrail은 직접 수행하거나 AWS 계정을 대신하여 수행한 API 직접 호출 및 관련 이벤트를 캡처하고 지정한 Amazon S3 버킷에 로그 파일을 전송합니다. 어떤 사용자 및 계정이 AWS를 직접적으로 호출했는지, 어떤 소스 IP 주소에 직접 호출이 이루어졌는지, 언제 직접 호출이 발생했는지 확인할 수 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

Amazon CloudWatch를 사용하여 MemoryDB 모니터링

원시 데이터를 수집하여 읽기 가능하고 실시간에 가까운 지표로 처리하는 CloudWatch를 사용하여 MemoryDB를 모니터링할 수 있습니다. 이러한 통계는 15개월간 보관되므로 기록 정보에 액세스하고 웹 애플리케이션 또는 서비스가 어떻게 실행되고 있는지 전체적으로 더 잘 파악할 수 있습니다. 특정 임곗값을 주시하다가 해당 임곗값이 충족될 때 알림을 전송하거나 조치를 취하도록 경보를 설정할 수도 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하세요.

다음 섹션에는 MemoryDB의 지표와 차원이 나열되어 있습니다.

주제

호스트 수준 지표

`AWS/MemoryDB` 네임스페이스에는 다음과 같이 개별 노드에 대한 호스트 수준 지표가 포함되어 있습니다.

참고

- MemoryDB에 대한 지표

지표	설명	단위
`CPUUtilization`	전체 호스트에 대한 CPU 사용률 비율입니다. Valkey 및 Redis OSS는 단일 스레드이므로 vCPU가 4개 이상인 노드에 대해 `EngineCPUUtilization` 지표를 모니터링하는 것이 좋습니다.	%

지표	설명	단위
FreeableMemory	호스트에서 사용 가능한 메모리의 양입니다. 이 숫자는 OS가 여유 공간으로 보고한 RAM의 메모리, 버퍼에서 나왔습니다.	바이트
NetworkBytesIn	호스트가 네트워크에서 읽어온 바이트 수입니다.	바이트
NetworkBytesOut	인스턴스가 모든 네트워크 인터페이스에서 보낸 바이트 수입니다.	바이트
NetworkPacketsIn	인스턴스가 모든 네트워크 인터페이스에서 받은 패킷 수입니다. 이 지표는 단일 인스턴스에서 수신 트래픽의 볼륨을 패킷 수 기준으로 식별합니다.	개수
NetworkPacketsOut	인스턴스가 모든 네트워크 인터페이스에서 보낸 패킷 수입니다. 이 지표는 단일 인스턴스에서 발신 트래픽의 볼륨을 패킷 수 기준으로 식별합니다.	개수
NetworkBandwidthInAllowanceExceeded	인바운드 집계 대역폭이 인스턴스의 최대값을 초과하여 형성된 패킷 수입니다.	개수
NetworkConntrackAllowanceExceeded	연결 추적이 인스턴스의 최대값을 초과하여 새 연결을 설정할 수 없기 때문에 형성된 패킷 수입니다. 이로 인해 인스턴스의 수신 또는 송신 트래픽에 대한 패킷 손실이 발생할 수 있습니다.	개수
NetworkBandwidthOutAllowanceExceeded	아웃바운드 집계 대역폭이 인스턴스의 최대값을 초과하여 형성된 패킷 수입니다.	개수
NetworkPacketsPerSecondAllowanceExceeded	양방향 PPS(packets per second)가 인스턴스의 최대값을 초과하여 형성된 패킷 수입니다.	개수
NetworkMaxBytesIn	1분마다 수신된 바이트의 초당 최대 버스트입니다.	바이트

지표	설명	단위
NetworkMaxBytesOut	1분마다 전송된 바이트의 초당 최대 버스트입니다.	바이트
NetworkMaxPacketsIn	1분마다 수신된 패킷의 초당 최대 버스트입니다.	개수
NetworkMaxPacketsOut	1분마다 전송된 패킷의 초당 최대 버스트입니다.	개수
SwapUsage	호스트에서 사용되는 스왑의 양입니다.	바이트

MemoryDB에 대한 지표

AWS/MemoryDB 네임스페이스에는 다음과 같은 지표가 포함됩니다.

ReplicationLag, EngineCPUUtilization, SuccessfulWriteRequestLatency, SuccessfulReadRequestLatency를 제외하고 이 지표는 Valkey 및 Redis OSS info 명령에서 파생됩니다. 각 지표는 노드 수준에서 계산됩니다.

INFO 명령의 전체 설명서는 INFO를 참조하세요.

또한 다음 섹션도 참조하세요.

- 호스트 수준 지표

지표	설명	단위
ActiveDefragHits	활성 조각 모음 프로세스에서 수행된 분당 값 재할당 수입니다. 이 지표는 INFO의 active_defrag_hits 통계에서 파생됩니다.	번호
AuthenticationFailures	AUTH 명령을 사용하여 실패한 인증 시도의 총 수입니다. 개별 인증 실패에 대한 자세한 내용은 ACL LOG 명령을 사용하여 확인할 수 있습니다. 무단 액세스 시도를 감지하려면 이에 대한 경보를 설정하는 것이 좋습니다.	개수

지표	설명	단위
BytesUsedForMemoryDB	데이터 세트, 버퍼 등을 포함하여 모든 목적을 위해 MemoryDB에서 할당한 전체 바이트 수.	바이트
	데이터 계층화를 사용하는 클러스터용 Dimension: Tier=SSD : SSD에서 사용된 총 바이트 수입니다.	바이트
	데이터 계층화를 사용하는 클러스터용 Dimension: Tier=Memory : 메모리에서 사용된 총 바이트 수입니다. INFO에 있는 used_memory 통계 값입니다.	바이트
BytesReadFromDisk	분당 디스크에서 읽은 총 바이트 수입니다. 데이터 계층화를 사용하는 클러스터에서만 지원됩니다.	바이트
BytesWrittenToDisk	분당 디스크에 쓴 총 바이트 수입니다. 데이터 계층화를 사용하는 클러스터에서만 지원됩니다.	바이트
CommandAuthorizationFailures	사용자가 호출 권한이 없는 명령을 실행한 실패한 시도의 총 수입니다. 개별 인증 실패에 대한 자세한 내용은 ACL LOG 명령을 사용하여 확인할 수 있습니다. 무단 액세스 시도를 감지하려면 이에 대한 경보를 설정하는 것이 좋습니다.	개수
CurrConnections	읽기 전용 복제본의 연결을 제외한 클라이언트 연결 수입니다. MemoryDB는 2~4개의 연결을 사용하여 각 사례의 클러스터를 모니터링합니다. 이 지표는 INFO의 connected_clients 통계에서 파생됩니다.	개수
CurrItems	캐시 항목 수입니다. 이 지표는 전체 키스페이스의 모든 키를 합산하여 keyspace 통계에서 파생됩니다.	개수

지표	설명	단위
	데이터 계층화를 사용하는 클러스터용 Dimension: Tier=Memory 입니다. 메모리에 있는 항목 수입니다.	개수
	데이터 계층화를 사용하는 클러스터용 Dimension: Tier=SSD (solid state drives)입니다. SSD의 항목 수입니다.	개수
DatabaseMemoryUsagePercentage	사용 중인 클러스터에 사용할 수 있는 메모리의 백분율입니다. 이 지표는 INFO의 used_memory/maxmemory 를 사용하여 계산됩니다.	%
DatabaseCapacityUsagePercentage	사용 중인 클러스터용 전체 데이터 용량의 백분율입니다. 데이터 계층형 인스턴스의 경우 지표는 (used_memory - mem_not_counted_for_evict + SSD used) / (maxmemory + SSD total capacity) 로 계산되며, 여기서 used_memory 및 maxmemory 는 INFO에서 가져옵니다. 다른 모든 경우 지표는 used_memory/maxmemory 를 사용하여 계산됩니다.	%
DB0AverageTTL	INFO 명령의 keyspace 통계에서 DBO의 avg_ttl을 표시합니다.	밀리초

지표	설명	단위
`EngineCPUUtilization`	Valkey 또는 Redis OSS 엔진 스레드의 CPU 사용률을 제공합니다. 이 엔진은 단일 스레드이므로 이 지표를 사용하여 프로세스 자체의 로드를 분석할 수 있습니다. `EngineCPUUtilization` 지표는 프로세스에 대한 보다 정확한 정보를 제공합니다. 이 지표를 `CPUUtilization` 지표와 함께 사용할 수 있습니다. `CPUUtilization` 은 다른 운영 체제 및 관리 프로세스를 포함하여 전체적인 서버 인스턴스의 CPU 사용률을 표시합니다. 4개 이상의 vCPU를 포함하는 대규모 노드 유형에는 `EngineCPUUtilization` 지표를 사용하여 조정 임곗값을 모니터링하고 설정하세요. **Note** MemoryDB 호스트에서는 백그라운드 프로세스가 관리형 데이터베이스 환경을 제공하기 위해 호스트를 모니터링합니다. 이러한 백그라운드 프로세스는 CPU 워크로드의 상당 부분을 차지할 수 있습니다. vCPU가 2개 이상인 대규모 호스트에서는 이 점이 중요하지 않습니다. 하지만 vCPU가 2개 이하인 소규모 호스트에 영향을 줄 수 있습니다. `EngineCPUUtilization` 지표만 모니터링하는 경우, 호스트가 Valkey 또는 Redis OSS 엔진의 높은 CPU 사용률과 백그라운드 모니터링 프로세스의 높은 CPU 사용률로 오버로드되는 상황을 인식하지 못합니다. 따라서 vCPU가 2개 이하인 호스트의 `CPUUtilization` 지표를 모니터링하는 것이 좋습니다.	%

지표	설명	단위
Evictions	maxmemory 제한으로 인해 제거된 키 수입니다. 이 지표는 INFO의 evicted_keys 통계에서 파생됩니다.	개수
IsPrimary	노드가 현재 샤드의 프라이머리 노드인지 여부를 나타냅니다. 이 지표는 0(프라이머리 노드 아님) 또는 1(프라이머리 노드임)일 수 있습니다.	개수
KeyAuthorizationFailures	사용자가 액세스 권한이 없는 키에 액세스한 실패한 시도의 총 수입니다. 개별 인증 실패에 대한 자세한 내용은 ACL LOG 명령을 사용하여 확인할 수 있습니다. 무단 액세스 시도를 감지하려면 이에 대한 경보를 설정하는 것이 좋습니다.	개수
KeyspaceHits	기본 사전의 성공한 읽기 전용 키 조회수입니다. 이 지표는 INFO의 keyspace_hits 통계에서 파생됩니다.	개수
KeyspaceMisses	기본 사전의 성공하지 못한 읽기 전용 키 조회 수입니다. 이 지표는 INFO의 keyspace_misses 통계에서 파생됩니다.	개수
KeysTracked	키 추적에서 추적 중인 키의 수로, tracking-table-max-keys 의 백분율로 표시됩니다. 키 추적은 클라이언트 측 캐싱을 지원하고 키가 수정된 경우, 클라이언트에 알리는 데 사용됩니다.	개수

지표	설명	단위
MaxReplicationThroughput	관찰된 최대 처리량입니다. 트래픽 버스트를 식별하기 위해 짧은 시간 간격으로 처리량을 샘플링합니다. 샘플링된 값의 최대값이 보고됩니다. 샘플링은 1분 간격으로 이루어집니다. 예를 들어 10ms 기간 동안 1MB의 데이터가 기록되는 경우 이 지표의 값은 100MBps가 됩니다. 쓰기 처리량 스로틀링으로 인해 이 지표가 100MBps를 초과할 때 쓰기 지연 시간이 더 길어질 수 있습니다.	초당 바이트
MemoryFragmentationRatio	Valkey 또는 Redis OSS 엔진의 메모리 할당의 효율성을 나타냅니다. 특정 임곗값은 다른 동작을 나타냅니다. 조각화를 1.0 이상으로 설정하는 것이 좋습니다. 이 수는 INFO의 mem_fragmentation_ratio statistic 에서 계산됩니다.	번호
MultiRegionClusterReplicationLag	MemoryDB Multi Region 클러스터에서는 MultiRegionClusterReplicationLag 이 리전 클러스터의 다중 AZ 트랜잭션 로그에 기록된 업데이트와 이 업데이트가 Multi Region 클러스터에 있는 다른 리전 클러스터의 프라이머리 노드에 기록된 시간 사이의 경과 시간을 측정합니다. 이 지표는 샤드 수준의 모든 소스 및 대상 리전 페어에 대해 내보내집니다.	밀리초
NewConnections	이 기간에 서버에서 허용된 총 연결 수입니다. 이 지표는 INFO의 total_connections_received 통계에서 파생됩니다.	개수
NumItemsReadFromDisk	분당 디스크에서 검색된 총 항목 수입니다. 데이터 계층화를 사용하는 클러스터에서만 지원됩니다.	개수
NumItemsWrittenToDisk	분당 디스크에 기록된 총 항목 수입니다. 데이터 계층화를 사용하는 클러스터에서만 지원됩니다.	개수

지표	설명	단위
PrimaryLinkHealthStatus	이 상태에는 0 또는 1의 두 가지 값이 있습니다. 값 0은 MemoryDB 프라이머리 노드의 데이터가 EC2의 Valkey 또는 Redis OSS와 동기화되지 않았음을 나타냅니다. 값 1은 데이터가 동기화되었음을 나타냅니다.	불린(Boolean)
Reclaimed	키 만료 이벤트 총 수입니다. 이 지표는 INFO의 `expired_keys` 통계에서 파생됩니다.	개수
ReplicationBytes	복제된 구성 노드의 경우, `ReplicationBytes` 가 프라이머리 노드에서 모든 복제본에 전송하는 바이트 수를 보고합니다. 이 지표는 클러스터에 대한 쓰기 부하를 나타냅니다. 이 지표는 INFO의 `master_repl_offset` 통계에서 파생됩니다.	바이트
ReplicationDelayedWriteCommands	동기 복제로 인해 지연된 쓰기 명령 수입니다. 네트워크 혼잡 또는 최대 복제 처리량 초과와 같은 다양한 요인으로 인해 복제가 지연될 수 있습니다.	개수
ReplicationLag	이 지표는 읽기 복제본으로 실행되는 노드에 한해 적용됩니다. 프라이머리 노드에서 변경 내용을 적용할 때 복제본에서 경과된 시간(초)을 나타냅니다.	초
SuccessfulWriteRequestLatency	성공적인 쓰기 요청의 지연 시간 유효한 통계: Average, Sum, Min, Max, Sample Count, p0과 p100 사이의 모든 백분위수. 샘플 수에는 성공적으로 실행된 명령만 포함됩니다. 사용 가능한 Valkey 7.2 이상.	마이크로초

지표	설명	단위
SuccessfulReadRequestLatency	성공적인 읽기 요청의 지연 시간 유효한 통계: Average, Sum, Min, Max, Sample Count, p0과 p100 사이의 모든 백분위수. 샘플 수에는 성공적으로 실행된 명령만 포함됩니다. 사용 가능한 Valkey 7.2 이상.	마이크로초
ErrorCount	지정된 기간 동안 실패한 총 명령 수입니다. 유효한 통계: Average, Sum, Min, Max	개수

다음은 info commandstats에서 파생된 몇 가지 유형의 명령 모음입니다. commandstats 섹션은 직접 호출 수를 포함하여 명령 유형에 따른 통계를 제공합니다.

사용 가능한 명령의 전체 목록은 명령을 참조하세요.

지표	설명	단위
EvalBasedCmds	EVAL 기반 명령의 총 명령 수입니다. 이 지표는 eval, evalsha를 합산하여 `commandstats` 통계에서 파생됩니다.	개수
GeoSpatialBasedCmds	지리 기반 명령의 총 명령 수입니다. 이 수는 `commandstats` 통계 자료에서 나왔습니다. 이 수는 모든 geo 유형의 명령(geoadd, geodist, geohash, geopos, georadius, georadius bymember)을 합산하여 계산됩니다.	개수
GetTypeCmds	read-only 유형 명령의 총 건수입니다. 이 수는 read-only 유형의 모든 명령(get, hget, scard, lrange 등)을 합산하여 `commandstats` 통계에서 나왔습니다.	개수
HashBasedCmds	해시 기반 명령 총 수입니다. 이 지표는 1개 이상의 해시(hget, hkeys, hvals, hdel 등)를 기반으	개수

지표	설명	단위
	로 실행되는 모든 명령을 합산하여 commandstats 통계에서 파생됩니다.	
HyperLogLogBasedCmds	HyperLogLog 기반 명령 총 건수입니다. 이 수는 pf 유형의 모든 명령(pfadd, pfcount, pfmerge 등)을 합산하여 commandstats 통계에서 나왔습니다.	개수
JsonBasedCmds	JSON 기반 명령 총 수입니다. 이 지표는 1개 이상의 JSON 문서 객체를 기반으로 실행되는 모든 명령을 합산하여 commandstats 통계에서 파생됩니다.	개수
KeyBasedCmds	키 기반 명령 총 수입니다. 이 수는 여러 데이터 구조(del, expire, rename 등) 상에서 1개 이상의 키에서 작동하는 모든 명령을 합산하여 commandstats 통계에서 나왔습니다.	개수
ListBasedCmds	목록 기반 명령 총 수입니다. 이 지표는 1개 이상의 목록(lindex, lrange, lpush, ltrim 등)을 기반으로 실행되는 모든 명령을 합산하여 commandstats 통계에서 파생됩니다.	개수
PubSubBasedCmds	pub/sub 기능의 명령 총 수입니다. 이 지표는 pub/sub 기능에 사용되는 모든 명령(psubscribe, publish, pubsub, punsubscribe, subscribe 및 unsubscribe)을 합산하여 commandstats 통계에서 파생됩니다.	개수
SearchBasedCmds	읽기 및 쓰기 명령을 포함한 총 보조 인덱스 및 검색 명령 수입니다. 이 지표는 보조 인덱스를 기반으로 실행되는 모든 검색 명령을 합산하여 commandstats 통계에서 파생됩니다.	개수

지표	설명	단위
SearchBasedGetCmds	보조 인덱스 및 검색 읽기 전용 명령의 총 수입니다. 이 지표는 모든 보조 인덱스 및 검색 가져오기 명령을 합산하여 commandstats 통계에서 파생됩니다.	개수
SearchBasedSetCmds	보조 인덱스 및 검색 쓰기 명령의 총 수입니다. 이 지표는 모든 보조 인덱스 및 검색 세트 명령을 합산하여 commandstats 통계에서 파생됩니다.	개수
SearchNumberOfIndexes	총 인덱스 수입니다.	개수
SearchNumberOfIndexedKeys	인덱싱된 키의 총 수	개수
SearchTotalIndexSize	모든 인덱스에서 사용하는 메모리(바이트)입니다.	바이트
SetBasedCmds	집합 기반 명령 총 수입니다. 이 지표는 1개 이상의 집합(scard, sdiff, sadd, sunion 등)을 기반으로 실행되는 모든 명령을 합산하여 commandstats 통계에서 파생됩니다.	개수
SetTypeCmds	write 유형의 총 명령 건수입니다. 이 수는 데이터(set, hset, sadd, lpop 등)에서 작동하는 모든 mutative 유형의 명령을 합산하여 commandstats 통계에서 나왔습니다.	개수
SortedSetBasedCmds	정렬된 집합 기반 명령 총 수입니다. 이 지표는 1개 이상의 정렬된 집합(zcount, zrange, zrank, zadd 등)을 기반으로 실행되는 모든 명령을 합산하여 commandstats 통계에서 파생됩니다.	개수

지표	설명	단위
`StringBasedCmds`	문자열 기반 명령 총 수입니다. 이 지표는 1개 이상의 문자열(strlen, setex, setrange 등)을 기반으로 실행되는 모든 명령을 합산하여 `commandstats` 통계에서 파생됩니다.	개수
`StreamBasedCmds`	총 스트림 기반 명령 수입니다. 이 지표는 1개 이상의 스트림 데이터 형식(xrange, xlen, xadd, xdel 등)을 기반으로 실행되는 모든 명령을 합산하여 `commandstats` 통계에서 파생됩니다.	개수

어떤 지표를 모니터링해야 합니까?

다음 CloudWatch 지표는 MemoryDB 성능에 대한 좋은 정보를 제공합니다. 대부분의 경우, 이러한 지표에 대해 성능 문제가 생기기 전에 수정 조치를 취할 수 있도록 CloudWatch 경보를 설정하는 것이 좋습니다.

모니터링할 지표

- CPUUtilization
- EngineCPUUtilization
- SwapUsage
- Evictions
- CurrConnections
- Memory
- Network
- Latency
- 복제

CPUUtilization

이는 백분율(%)로 보고된 호스트 수준 지표입니다. 자세한 내용은 호스트 수준 지표 섹션을 참조하세요.

2vCPU 이하의 작은 노드 유형에는 `CPUUtilization` 지표를 사용하여 워크로드를 모니터링하세요.

일반적으로, 사용 가능한 CPU의 90%로 임곗값을 설정하는 것이 좋습니다. Valkey 및 Redis OSS는 단일 스레드이기 때문에 실제 임곗값은 노드 총용량의 일부로 계산해야 합니다. 2개의 코어가 있는 노드 유형을 사용하는 경우를 예로 들어보겠습니다. 이 경우, CPUUtilization의 임곗값은 90/2 또는 45%입니다. 노드 유형에 있는 코어(vCPU)의 수를 확인하려면 MemoryDB 요금을 참조하세요.

사용 중인 노드에 있는 코어 개수에 따라 임곗값을 결정해야 합니다. 이 임곗값을 초과하고, 주된 워크로드가 읽기 요청에서 비롯되는 경우에는 읽기 전용 복제본을 추가하여 클러스터를 스케일 아웃합니다. 기본 워크로드가 쓰기 요청에서 발생하는 경우, 샤드를 추가하여 쓰기 워크로드를 프라이머리 노드 전체에 더 많이 배포합니다.

> **Tip**
>
> 호스트 수준 지표 `CPUUtilization`을 사용하는 대신, Valkey 또는 Redis OSS 엔진 코어의 사용률을 보고하는 지표 `EngineCPUUtilization`을 사용할 수 있습니다. 노드에서 이 지표를 사용할 수 있는지 확인하고 자세한 내용을 보려면 MemoryDB 지표를 참조하세요.

4vCPU 이상의 대규모 노드 유형에는 Valkey 또는 Redis OSS 엔진 코어의 대한 사용률을 보고하는 `EngineCPUUtilization` 지표를 사용할 수 있습니다. 노드에서 이 지표를 사용할 수 있는지 확인하고 자세한 내용을 보려면 MemoryDB 지표를 참조하세요.

EngineCPUUtilization

4vCPU 이상의 대규모 노드 유형에는 Valkey 또는 Redis OSS 엔진 코어의 대한 사용률을 보고하는 `EngineCPUUtilization` 지표를 사용할 수 있습니다. 노드에서 이 지표를 사용할 수 있는지 확인하고 자세한 내용을 보려면 MemoryDB 지표를 참조하세요.

SwapUsage

이는 바이트로 보고된 호스트 수준 지표입니다. 자세한 내용은 호스트 수준 지표 섹션을 참조하세요.

`FreeableMemory` CloudWatch 지표가 0에 가깝거나(즉, 100MB 미만) `SwapUsage` 지표가 `FreeableMemory` 지표보다 큰 경우 노드가 메모리 부족 상태일 수 있습니다.

Evictions

이것은 엔진 지표입니다. 애플리케이션 요구 사항에 따라 이 지표에 대한 경보 임곗값을 결정하는 것이 좋습니다.

CurrConnections

이것은 엔진 지표입니다. 애플리케이션 요구 사항에 따라 이 지표에 대한 경보 임곗값을 결정하는 것이 좋습니다.

CurrConnections 개수가 증가하는 것은 애플리케이션에 문제가 있음을 나타내며 이 문제를 해결하려면 애플리케이션 동작을 확인해야 합니다.

Memory

메모리는 Valkey 및 Redis OSS의 핵심적인 측면입니다. 데이터 손실을 방지하고 데이터 집합의 향후 증가를 수용하려면 클러스터의 메모리 사용률을 파악할 필요가 있습니다. 노드의 메모리 사용률에 대한 통계는 INFO 명령의 메모리 섹션에서 확인할 수 있습니다.

Network

클러스터의 네트워크 대역폭 용량을 결정하는 요인 중 하나는 선택한 노드 유형입니다. 노드의 네트워크 용량에 대한 자세한 내용은 Amazon MemoryDB 요금 섹션을 참조하세요.

Latency

지연 시간 지표 `SuccessfulWriteRequestLatency` 및 `SuccessfulReadRequestLatency`는 Valkey 엔진의 MemoryDB가 요청에 응답하는 데 걸리는 총 시간을 측정합니다.

> **Note**
>
> Valkey 클라이언트에서 CLIENT REPLY가 활성화된 상태에서 Valkey 파이프라이닝을 사용할 때 `SuccessfulWriteRequestLatency` 및 `SuccessfulReadRequestLatency` 지표에 대한 팽창된 값이 발생할 수 있습니다. Valkey 파이프라이닝은 각 개별 명령에 대한 응답을 기다리지 않고 한 번에 여러 명령을 실행하여 성능을 개선하는 기법입니다. 값이 부풀어 오르지 않도록 하려면 CLIENT REPLY OFF를 사용하여 명령을 파이프라인하도록 Redis 클라이언트를 구성하는 것이 좋습니다.

복제

복제되는 데이터의 볼륨은 `ReplicationBytes` 지표를 통해 확인할 수 있습니다. 복제 용량 처리량에 대해 `MaxReplicationThroughput`을 모니터링할 수 있습니다. 최대 복제 용량 처리량에 도달하면 샤드를 더 추가하는 것이 좋습니다.

`ReplicationDelayedWriteCommands`은(는) 또한 워크로드가 최대 복제 용량 처리량을 초과하는지 여부를 나타낼 수도 있습니다. MemoryDB 복제에 대한 자세한 내용은 MemoryDB 복제 이해를 참조하세요.

지표 통계 및 기간 선택

CloudWatch에서 각 지표의 통계 및 기간을 선택하도록 허용하며 모든 조합이 유용한 것은 아닙니다. 예를 들어, CPUUtilization에 대한 Average, Minimum 및 Maximum 통계는 유용하지만 Sum 통계는 유용하지 않습니다.

모든 MemoryDB 샘플은 각 개별 노드에 대해 60초 동안 게시됩니다. 60초 기간 동안 노드 지표는 단일 샘플만 포함할 수 있습니다.

CloudWatch 지표 모니터링

MemoryDB와 CloudWatch가 서로 통합되어 있어서 다양한 지표를 수집할 수 있습니다. CloudWatch를 사용하여 이러한 지표를 모니터링할 수 있습니다.

> **Note**
>
> 다음 예제를 실행하려면 CloudWatch 명령줄 도구가 필요합니다. CloudWatch에 대해 자세한 내용을 알아보고 개발자 도구를 다운로드하려면 CloudWatch 제품 페이지를 참조하세요.

다음 절차에서는 CloudWatch를 사용하여 지난 시간 클러스터의 스토리지 공간 통계를 수집하는 방법을 보여줍니다.

> **Note**
>
> 아래 예제에 나온 `StartTime` 및 `EndTime` 값은 설명을 돕기 위해 지정되었습니다. 노드의 올바른 시작 및 종료 시간 값으로 대체해야 합니다.

MemoryDB 제한에 대한 자세한 내용은 MemoryDB의 AWS서비스 제한을 참조하세요.

CloudWatch 지표 모니터링(콘솔)

클러스터의 CPU 사용률 통계를 수집하려면

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 지표를 확인할 노드를 선택합니다.

> **Note**
>
> 20개보다 많은 노드를 선택하면 콘솔에 지표가 표시되지 않습니다.

a. AWS Management Console의 클러스터 페이지에서 하나 이상의 클러스터 이름을 클릭합니다.

   클러스터의 세부 정보 페이지가 나타납니다.

b. 창 맨 위의 [Nodes] 탭을 클릭합니다.

c. 세부 정보 창의 [Nodes] 탭에서 지표를 확인할 노드를 선택합니다.

   사용 가능한 CloudWatch 지표 목록이 콘솔 창 하단에 나타납니다.

d. [CPU Utilization] 지표를 클릭합니다.

   선택한 지표가 표시된 CloudWatch 콘솔이 열립니다. 통계 및 기간 드롭다운 목록 상자와 시간 범위 탭을 사용하여 표시되는 지표를 변경할 수 있습니다.

CloudWatch CLI를 사용하여 CloudWatch 지표 모니터링

클러스터의 CPU 사용률 통계를 수집하려면

- CloudWatch 명령 aws cloudwatch get-metric-statistics을(를) 다음 파라미터와 함께 사용합니다. 시작 및 종료 시간은 예제와 같이 표시되며 적절한 시작 및 종료 시간으로 대체해야 합니다.

  Linux, macOS, Unix의 경우:

```
aws cloudwatch get-metric-statistics CPUUtilization \
    --dimensions=ClusterName=mycluster,NodeId=0002" \
    --statistics=Average \
    --namespace="AWS/MemoryDB" \
    --start-time 2013-07-05T00:00:00 \
    --end-time 2013-07-06T00:00:00 \
    --period=60
```

  Windows의 경우:

```
mon-get-stats CPUUtilization ^
```

```
    --dimensions=ClusterName=mycluster,NodeId=0002" ^
    --statistics=Average ^
    --namespace="AWS/MemoryDB" ^
    --start-time 2013-07-05T00:00:00 ^
    --end-time 2013-07-06T00:00:00 ^
    --period=60
```

CloudWatch API를 사용하여 CloudWatch 지표 모니터링

클러스터의 CPU 사용률 통계를 수집하려면

- CloudWatch API `GetMetricStatistics`를 다음 파라미터와 함께 호출합니다. 시작 및 종료 시간은 예제와 같이 표시되며 적절한 시작 및 종료 시간으로 대체해야 합니다.
  - `Statistics.member.1=Average`
  - `Namespace=AWS/MemoryDB`
  - `StartTime=2013-07-05T00:00:00`
  - `EndTime=2013-07-06T00:00:00`
  - `Period=60`
  - `MeasureName=CPUUtilization`
  - `Dimensions=ClusterName=mycluster,NodeId=0002`

  Example

```
http://monitoring.amazonaws.com/
    ?SignatureVersion=4
    &Action=GetMetricStatistics
    &Version=2014-12-01
    &StartTime=2013-07-16T00:00:00
    &EndTime=2013-07-16T00:02:00
    &Period=60
    &Statistics.member.1=Average
    &Dimensions.member.1="ClusterName=mycluster"
    &Dimensions.member.2="NodeId=0002"
    &Namespace=Amazon/memorydb
    &MeasureName=CPUUtilization
    &Timestamp=2013-07-07T17%3A48%3A21.746Z
    &AWS;AccessKeyId=<&AWS; Access Key ID>
```

```
&Signature=<Signature>
```

MemoryDB 이벤트 모니터링

클러스터에서 중요 이벤트가 발생하면 MemoryDB는 특정 Amazon SNS 주제에 알림을 전송합니다. 이러한 예에는 노드 추가 실패, 노드 추가 성공, 보안 그룹 수정 등이 있습니다. 주요 이벤트를 모니터링하면 클러스터의 현재 상태를 파악할 수 있으며, 이벤트에 따라 교정 작업을 수행할 수도 있습니다.

주제

- [MemoryDB Amazon SNS 알림 관리]
- [MemoryDB 이벤트 보기]
- [이벤트 알림 및 Amazon SNS]

MemoryDB Amazon SNS 알림 관리

Amazon Simple Notification Service(SNS)를 사용하여 중요한 클러스터 이벤트에 대해 알림을 보내도록 MemoryDB를 구성할 수 있습니다. 이러한 예에서는 Amazon SNS 항목의 Amazon 리소스 이름(ARN)으로 클러스터를 구성하여 알림을 받습니다.

> **Note**
> 이 항목에서는 Amazon SNS에 가입했으며 Amazon SNS 주제를 설정 및 구독했다고 가정합니다. 이렇게 하는 방법에 대한 정보는 Amazon Simple Notification Service 개발자 안내서를 참조하세요.

Amazon SNS 주제 추가

다음 섹션은 AWS 콘솔, AWS CLI 또는 MemoryDB API를 사용하여 Amazon SNS 주제를 추가하는 방법을 보여줍니다.

Amazon SNS 주제 추가(콘솔)

다음 절차는 클러스터에 대해 Amazon SNS 주제를 추가하는 방법을 보여줍니다.

> **Note**
>
> 이 프로세스는 Amazon SNS 주제를 수정하는 데에도 사용할 수 있습니다.

클러스터에 대해 Amazon SNS 주제를 추가 또는 수정하려면(콘솔)

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 클러스터에서 Amazon SNS 주제 ARN을 추가 또는 수정할 클러스터를 선택합니다.
3. 수정을 선택합니다.
4. 클러스터 수정의 SNS 알림에 대한 주제에서 추가하려는 SNS 주제를 선택하거나 수동 ARN 입력을 선택하고 Amazon SNS 주제의 ARN을 입력합니다.
5. 수정을 선택합니다.

Amazon SNS 주제 추가(AWS CLI)

클러스터에 대해 Amazon SNS 주제를 추가 또는 수정하려면 AWS CLI 명령 `update-cluster`를 사용합니다.

다음 코드 예제는 Amazon SNS 주제 arn을 my-cluster에 추가합니다.

Linux, macOS, Unix의 경우:

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --sns-topic-arn arn:aws:sns:us-east-1:565419523791:memorydbNotifications
```

Windows의 경우:

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --sns-topic-arn arn:aws:sns:us-east-1:565419523791:memorydbNotifications
```

자세한 내용은 UpdateCluster를 참조하세요.

Amazon SNS 주제 추가(MemoryDB API)

클러스터에 대해 Amazon SNS 주제를 추가 또는 업데이트하려면 다음 파라미터와 함께 `UpdateCluster` 작업을 호출합니다.

- `ClusterName=my-cluster`
- `SnsTopicArn=arn%3Aaws%3Asns%3Aus-east-1%3A565419523791%3AmemorydbNotifications`

클러스터에 대해 Amazon SNS 주제를 추가 또는 업데이트하려면 `UpdateCluster` 작업을 직접적으로 호출합니다.

자세한 내용은 UpdateCluster를 참조하세요.

Amazon SNS 알림 활성화 및 비활성화

클러스터에 대해 알림을 켜거나 끌 수 있습니다. 다음 절차는 Amazon SNS 알림을 비활성화하는 방법을 보여줍니다.

Amazon SNS 알림 활성화 및 비활성화(콘솔)

AWS Management 콘솔을 사용하여 Amazon SNS 알림을 비활성화하려면

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 알림을 수정할 클러스터의 이름 왼쪽에 있는 라디오 버튼을 선택합니다.
3. 수정을 선택합니다.
4. 클러스터 수정의 SNS 알림에 대한 주제에서 알림 비활성화를 선택합니다.
5. 수정을 선택합니다.

Amazon SNS 알림 활성화 및 비활성화(AWS CLI)

Amazon SNS 알림을 비활성화하려면 다음 파라미터와 함께 `update-cluster` 명령을 사용합니다.

Linux, macOS, Unix의 경우:

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --sns-topic-status inactive
```

Windows의 경우:

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --sns-topic-status inactive
```

Amazon SNS 알림 활성화 및 비활성화(MemoryDB API)

Amazon SNS 알림을 비활성화하려면 다음 파라미터와 함께 `UpdateCluster` 작업을 호출합니다.

- `ClusterName=my-cluster`
- `SnsTopicStatus=inactive`

이 호출은 다음과 비슷한 출력을 반환합니다.

Example

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=UpdateCluster
    &ClusterName=my-cluster
    &SnsTopicStatus=inactive
    &Version=2021-01-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210801T220302Z
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

MemoryDB 이벤트 보기

MemoryDB 로그는 클러스터, 보안 그룹 및 파라미터 그룹과 관련된 이벤트를 기록합니다. 여기에는 이벤트 날짜 및 시간, 이벤트의 원본 이름 및 원본 유형, 이벤트 설명 등의 정보가 포함됩니다. MemoryDB 콘솔, AWS CLI `describe-events` 명령 또는 MemoryDB API 작업 `DescribeEvents`을 사용하여 로그에서 이벤트를 쉽게 검색할 수 있습니다.

다음 절차는 지난 24시간(1440분) 동안의 모든 MemoryDB 이벤트를 보는 방법을 보여줍니다.

MemoryDB 이벤트 보기(콘솔)

다음 절차는 MemoryDB 콘솔을 사용하여 이벤트를 표시합니다.

MemoryDB 콘솔을 사용하여 이벤트를 보려면

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 이벤트를 선택합니다.

   사용 가능한 모든 이벤트가 나열된 이벤트 화면이 나타납니다. 이벤트 화면에서 목록의 각 행은 이벤트 하나를 나타내며 이벤트 소스, 이벤트 유형(클러스터, parameter-group, acl, 보안 그룹 또는 서브넷 그룹), 이벤트의 GMT 시간 및 이벤트 설명을 표시합니다.

   [Filter]를 사용하여 이벤트 목록에서 모든 이벤트를 볼지 특정 유형의 이벤트만 볼지를 지정할 수 있습니다.

MemoryDB 이벤트 보기(AWS CLI)

AWS CLI을(를) 사용하여 MemoryDB 이벤트의 목록을 생성하려면 `describe-events` 명령을 사용합니다. 선택적 파라미터를 사용하여 나열된 이벤트의 유형, 나열된 이벤트의 기간, 나열할 이벤트의 최대 수 등을 제어할 수 있습니다.

다음 코드는 최대 40개의 클러스터 이벤트를 나열합니다.

```
aws memorydb describe-events --source-type cluster --max-results 40
```

다음 코드는 지난 24시간(1440분) 동안의 모든 이벤트를 나열합니다.

```
aws memorydb describe-events --duration 1440
```

`describe-events` 명령의 출력은 다음과 같습니다.

```
{
    "Events": [
        {
            "Date": "2021-03-29T22:17:37.781Z",
            "Message": "Added node 0001 in Availability Zone us-east-1a",
            "SourceName": "memorydb01",
            "SourceType": "cluster"
        },
        {
            "Date": "2021-03-29T22:17:37.769Z",
            "Message": "cluster created",
            "SourceName": "memorydb01",
            "SourceType": "cluster"
        }
    ]
}
```

사용 가능한 파라미터 및 허용된 파라미터 값과 같은 자세한 내용은 [describe-events]를 참조하세요.

MemoryDB 이벤트 보기(MemoryDB API)

MemoryDB API를 사용하여 MemoryDB 이벤트의 목록을 생성하려면 `DescribeEvents` 작업을 사용합니다. 선택적 파라미터를 사용하여 나열된 이벤트의 유형, 나열된 이벤트의 기간, 나열할 이벤트의 최대 수 등을 제어할 수 있습니다.

다음 코드는 40개의 최신 클러스터 이벤트를 나열합니다.

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeEvents
   &MaxResults=40
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &SourceType=cluster
   &Timestamp=20210802T192317Z
   &Version=2021-01-01
   &X-Amz-Credential=<credential>
```

다음 코드는 지난 24시간(1440분) 동안의 클러스터 이벤트를 나열합니다.

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeEvents
   &Duration=1440
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &SourceType=cluster
   &Timestamp=20210802T192317Z
   &Version=2021-01-01
   &X-Amz-Credential=<credential>
```

위 작업을 통해 다음과 비슷한 출력이 생성되어야 합니다.

```
<DescribeEventsResponse xmlns="http://memory-db.us-east-1.amazonaws.com/
doc/2021-01-01/">
    <DescribeEventsResult>
        <Events>
            <Event>
                <Message>cluster created</Message>
                <SourceType>cluster</SourceType>
                <Date>2021-08-02T18:22:18.202Z</Date>
                <SourceName>my-memorydb-primary</SourceName>
            </Event>

 (...output omitted...)

        </Events>
    </DescribeEventsResult>
    <ResponseMetadata>
        <RequestId>e21c81b4-b9cd-11e3-8a16-7978bb24ffdf</RequestId>
    </ResponseMetadata>
</DescribeEventsResponse>
```

사용 가능한 파라미터 및 허용된 파라미터 값과 같은 자세한 내용은 `DescribeEvents`를 참조하세요.

이벤트 알림 및 Amazon SNS

MemoryDB는 클러스터에서 중요 이벤트가 발생하는 경우, Amazon Simple Notification Service(SNS)를 사용하여 메시지를 게시할 수 있습니다. 이 기능은 클러스터의 개별 노드 엔드포인트에 연결된 클라이언트 머신의 서버 목록을 새로 고침하는 데 사용될 수 있습니다.

> **Note**
>
> 이용 요금 정보 및 Amazon SNS 설명서 링크 등 Amazon Simple Notification Service(SNS)에 대한 자세한 내용은 Amazon SNS 제품 페이지를 참조하세요.

알림은 지정된 Amazon SNS 주제에 대해 게시됩니다. 다음은 알림에 대한 요구 사항입니다.

- MemoryDB 알림에 대해 주제 하나만 구성할 수 있습니다.
- Amazon SNS 주제를 소유한 AWS 계정은 알림이 활성화된 클러스터를 소유하는 계정과 동일해야 합니다.

MemoryDB 이벤트

다음 MemoryDB 이벤트는 Amazon SNS 알림을 트리거합니다.

Event Name	메시지	설명
MemoryDB:AddNodeComplete	`"Modified number of nodes from %d to %d"`	노드가 클러스터에 추가되었고 사용할 준비가 되어 있습니다.
무료 IP 주소가 부족함으로 인한 MemoryDB:AddNodeFailed	`"Failed to modify number of nodes from %d to %d due to insufficient free IP addresses"`	사용 가능한 IP 주소가 충분하지 않아 노드를 추가하지 못했습니다.
MemoryDB:ClusterParametersChanged	`"Updated parameter group for the cluster"` 생성 시 `"Updated to use a ParameterGroup %s"` 도 전송합니다.	하나 이상의 클러스터 파라미터가 변경되었습니다.
MemoryDB:ClusterProvisioningComplete	`"Cluster created."`	클러스터의 프로비저닝이 완료되어 클러스터에 있는 노드를 사용할 수 있습니다.

Event Name	메시지	설명
호환되지 않는 네트워크 상태로 인한 MemoryDB:ClusterProvisioningFailed	`"Failed to create cluster due to incompatible network state. %s"`	존재하지 않는 Virtual Private Cloud(VPC)에서 새로운 클러스터를 실행하려고 시도했습니다.
MemoryDB:ClusterRestoreFailed	`"Restore from %s failed for node %s. %s"`	MemoryDB가 클러스터를 스냅샷 데이터로 채우지 못했습니다. Amazon S3에 스냅샷 파일이 없거나 해당 파일에 대한 권한이 잘못되었기 때문일 수 있습니다. 클러스터를 설명할 경우, 상태는 `restore-failed` 입니다. 클러스터를 삭제하고 다시 시작해야 합니다. 자세한 내용은 외부에서 생성된 스냅샷으로 새 클러스터 시드 섹션을 참조하세요.
MemoryDB:ClusterScalingComplete	`"Succeeded applying modification to node type to %s."`	클러스터의 스케일 업이 성공적으로 완료되었습니다.
MemoryDB:ClusterScalingFailed	`"Failed applying modification to node type to %s."`	클러스터에 대한 스케일 업 작업이 실패했습니다.

Event Name	메시지	설명
MemoryDB:NodeReplaceStarted	"Recovering node %s"	MemoryDB가 노드를 실행하는 호스트 성능이 저하되었거나 연결되지 않음을 감지하여 노드 교체를 시작했습니다. **Note** 교체된 노드의 DNS 항목은 변경되지 않습니다. 대부분의 경우에 이 이벤트가 발생할 때 클라이언트의 서버 목록을 새로 고침하지 않아도 됩니다. 하지만 일부 클라이언트 라이브러리는 MemoryDB가 노드를 교체한 후에도 캐시 노드 사용을 중단할 수 있습니다. 이 경우에 애플리케이션은 이 이벤트가 발생할 때 서버 목록을 새로 고침해야 합니다.

Event Name	메시지	설명
MemoryDB:NodeReplaceComplete	"Finished recovery for node %s"	MemoryDB가 노드를 실행하는 호스트 성능이 저하되었거나 연결되지 않음을 감지하여 노드 교체를 완료했습니다. **Note** 교체된 노드의 DNS 항목은 변경되지 않습니다. 대부분의 경우에 이 이벤트가 발생할 때 클라이언트의 서버 목록을 새로 고침하지 않아도 됩니다. 하지만 일부 클라이언트 라이브러리는 MemoryDB가 노드를 교체한 후에도 캐시 노드 사용을 중단할 수 있습니다. 이 경우에 애플리케이션은 이 이벤트가 발생할 때 서버 목록을 새로 고침해야 합니다.
MemoryDB:CreateClusterComplete	"Cluster created"	클러스터가 성공적으로 생성되었습니다.
MemoryDB:CreateClusterFailed	"Failed to create cluster due to unsuccessful creation of its node(s)." ‘and’"Deleting all nodes belonging to this cluster."	클러스터가 생성되지 않았습니다.

Event Name	메시지	설명
MemoryDB:DeleteClusterComplete	`"Cluster deleted."`	클러스터 및 연결된 모든 노드 삭제를 완료했습니다.
MemoryDB:FailoverComplete	`"Failover to replica node %s completed"`	복제본 노드에 대한 장애 조치가 성공했습니다.
MemoryDB:NodeReplacementCanceled	`"The replacement of node %s which was scheduled during the maintenance window from start time: %s, end time: %s has been canceled"`	교체가 예약되어 있는 클러스터의 노드가 더 이상 교체 예약이 되지 않습니다.
MemoryDB:NodeReplacementRescheduled	`"The replacement in maintenance window for node %s has been re-scheduled from previous start time: %s, previous end time: %s to new start time: %s, new end time: %s"`	이전에 교체가 예약되어 있는 클러스터의 노드가 알림에 설명된 새 기간 동안 교체가 예약됩니다. 수행할 수 있는 작업에 대한 자세한 내용은 노드 교체 섹션을 참조하세요.
MemoryDB:NodeReplacementScheduled	`"The node %s is scheduled for replacement during the maintenance window from start time: %s to end time: %s"`	클러스터의 노드가 알림에 설명된 기간 동안 교체가 예약됩니다. 수행할 수 있는 작업에 대한 자세한 내용은 노드 교체 섹션을 참조하세요.
MemoryDB:RemoveNodeComplete	`"Removed node %s"`	노드가 클러스터에서 제거되었습니다.
MemoryDB:SnapshotComplete	`"Snapshot %s succeeded for node %s"`	스냅샷이 성공적으로 완료되었습니다.

Event Name	메시지	설명
MemoryDB:SnapshotFailed	`"Snapshot %s failed for node %s"`	스냅샷이 실패했습니다. 원인에 대한 자세한 내용은 클러스터의 이벤트를 참조하세요. 스냅샷을 설명할 경우, [DescribeSnapshots](#)를 참조하세요. 상태는 `failed`입니다.

AWS CloudTrail을 사용하여 MemoryDB API 직접 호출 로깅

MemoryDB는 MemoryDB에서 사용자, 역할 또는 AWS 서비스가 수행한 작업에 대한 레코드를 제공하는 서비스인 AWS CloudTrail과 통합됩니다. CloudTrail은 MemoryDB 콘솔의 직접 호출, MemoryDB API 작업에 대한 코드 직접 호출을 포함하여 MemoryDB에 대한 모든 API 직접 호출을 이벤트로 캡처합니다. 추적을 생성하면 MemoryDB 이벤트를 포함한 CloudTrail 이벤트를 지속적으로 Amazon S3 버킷에 배포할 수 있습니다. 추적을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 MemoryDB에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

CloudTrail의 MemoryDB 정보

CloudTrail은 계정 생성 시 AWS 계정에서 활성화됩니다. MemoryDB에서 활동이 발생하면 해당 활동이 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에서 최신 이벤트를 확인, 검색 및 다운로드할 수 있습니다. 자세한 설명은 CloudTrail 이벤트 기록으로 이벤트 보기를 참조하세요.

MemoryDB에 대한 이벤트를 포함하여 AWS 계정에 이벤트를 지속적으로 기록하려면 추적을 생성합니다. 트레일을 통해 CloudTrail은 로그 파일을 Amazon S3 버킷으로 전송할 수 있습니다. 콘솔에서 트레일을 생성하면 기본적으로 모든 리전에 트레일이 적용됩니다. 추적은 AWS파티션에 있는 모든 리전의 이벤트를 로깅하고 지정된 Amazon S3 버킷으로 로그 파일을 전송합니다. 추가적으로, CloudTrail 로그에서 수집된 이벤트 데이터를 추가 분석 및 처리하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음 자료를 참조하세요.

- 트레일 생성 개요
- CloudTrail 지원 서비스 및 통합

- CloudTrail에서 Amazon SNS 알림 구성
- 여러 리전으로부터 CloudTrail 로그 파일 받기 및 여러 계정으로부터 CloudTrail 로그 파일 받기

MemoryDB의 모든 작업은 CloudTrail에서 로깅됩니다. 예를 들어 `CreateCluster`, `DescribeClusters`, `UpdateCluster` 작업을 직접 호출하면 CloudTrail 로그 파일에 항목이 생성됩니다.

모든 이벤트 또는 로그 항목에는 요청을 생성했던 사용자에 대한 정보가 포함됩니다. ID 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 IAM 사용자 보안 인증 정보로 했는지 여부.
- 역할 또는 페더레이션 사용자의 임시 보안 인증을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청했는지.

자세한 설명은 CloudTrail userIdentity 요소를 참조하세요.

MemoryDB 로그 파일 항목에 대한 이해

추적이란 지정한 S3 버킷에 이벤트를 로그 파일로 입력할 수 있게 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스의 단일 요청을 나타내며 요청된 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보를 포함하고 있습니다. CloudTrail 로그 파일은 퍼블릭 API 직접 호출에 대한 순서 지정된 스택 추적이 아니기 때문에 특정 순서로 표시되지 않습니다.

다음은 `CreateCluster` 작업을 보여 주는 CloudTrail 로그 항목이 나타낸 예제입니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EKIAUAXQT3SWDEXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/john",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "john"
    },
    "eventTime": "2021-07-10T17:56:46Z",
    "eventSource": "memorydb.amazonaws.com",
    "eventName": "CreateCluster",
    "awsRegion": "us-east-1",
```

```
    "sourceIPAddress": "192.0.2.01",
    "userAgent": "aws-cli/2.2.29 Python/3.9.6 Darwin/19.6.0 source/x86_64 prompt/off
 command/memorydb.create-cluster",
    "requestParameters": {
        "clusterName": "memorydb-cluster",
        "nodeType": "db.r6g.large",
        "subnetGroupName": "memorydb-subnet-group",
        "aCLName": "open-access"
    },
    "responseElements": {
        "cluster": {
            "name": "memorydb-cluster",
            "status": "creating",
            "numberOfShards": 1,
            "availabilityMode": "MultiAZ",
            "clusterEndpoint": {
                "port": 6379
            },
            "nodeType": "db.r6g.large",
            "engineVersion": "6.2",
            "enginePatchVersion": "6.2.6",
            "parameterGroupName": "default.memorydb-redis6",
            "parameterGroupStatus": "in-sync",
            "subnetGroupName": "memorydb-subnet-group",
            "tLSEnabled": true,
            "aRN": "arn:aws:memorydb:us-east-1:123456789012:cluster/memorydb-cluster",
            "snapshotRetentionLimit": 0,
            "maintenanceWindow": "tue:06:30-tue:07:30",
            "snapshotWindow": "09:00-10:00",
            "aCLName": "open-access",
            "dataTiering": "false",
            "autoMinorVersionUpgrade": true
        }
    },
    "requestID": "506fc951-9ae2-42bb-872c-98028dc8ed11",
    "eventID": "2ecf3dc3-c931-4df0-a2b3-be90b596697e",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}
```

다음은 `DescribeClusters` 작업을 설명하는 CloudTrail 로그 항목을 보여 주는 예시입니다. 모든 MemoryDB Describe 및 List 직접 호출(`Describe*` 및 `List*`)에 대해 `responseElements` 섹션이 제거되고 `null`로 표시됩니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EKIAUAXQT3SWDEXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/john",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "john"
    },
    "eventTime": "2021-07-10T18:39:51Z",
    "eventSource": "memorydb.amazonaws.com",
    "eventName": "DescribeClusters",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.01",
    "userAgent": "aws-cli/2.2.29 Python/3.9.6 Darwin/19.6.0 source/x86_64 prompt/off
 command/memorydb.describe-clusters",
    "requestParameters": {
        "maxResults": 50,
        "showShardDetails": true
    },
    "responseElements": null,
    "requestID": "5e831993-52bb-494d-9bba-338a117c2389",
    "eventID": "32a3dc0a-31c8-4218-b889-1a6310b7dd50",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}
```

다음 예제는 `UpdateCluster` 작업을 기록하는 CloudTrail 로그 항목을 보여줍니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EKIAUAXQT3SWDEXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/john",
```

```
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "john"
    },
    "eventTime": "2021-07-10T19:23:20Z",
    "eventSource": "memorydb.amazonaws.com",
    "eventName": "UpdateCluster",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.01",
    "userAgent": "aws-cli/2.2.29 Python/3.9.6 Darwin/19.6.0 source/x86_64 prompt/off
 command/memorydb.update-cluster",
    "requestParameters": {
        "clusterName": "memorydb-cluster",
        "snapshotWindow": "04:00-05:00",
        "shardConfiguration": {
            "shardCount": 2
        }
    },
    "responseElements": {
        "cluster": {
            "name": "memorydb-cluster",
            "status": "updating",
            "numberOfShards": 2,
            "availabilityMode": "MultiAZ",
            "clusterEndpoint": {
                "address": "clustercfg.memorydb-cluster.cde8da.memorydb.us-
east-1.amazonaws.com",
                "port": 6379
            },
            "nodeType": "db.r6g.large",
            "engineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "parameterGroupName": "default.memorydb-redis6",
            "parameterGroupStatus": "in-sync",
            "subnetGroupName": "memorydb-subnet-group",
            "tLSEnabled": true,
            "aRN": "arn:aws:memorydb:us-east-1:123456789012:cluster/memorydb-cluster",
            "snapshotRetentionLimit": 0,
            "maintenanceWindow": "tue:06:30-tue:07:30",
            "snapshotWindow": "04:00-05:00",
            "autoMinorVersionUpgrade": true,
            "DataTiering": "false"
        }
    },
```

```
    "requestID": "dad021ce-d161-4365-8085-574133afab54",
    "eventID": "e0120f85-ab7e-4ad4-ae78-43ba15dee3d8",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}
```


다음은 `CreateUser` 작업을 설명하는 CloudTrail 로그 항목을 보여 주는 예시입니다. 민감한 데이터를 포함하는 MemoryDB 직접 호출의 경우 아래 `requestParameters` 섹션에 표시된 대로 해당 CloudTrail 이벤트에서 관련 데이터가 삭제된다는 점에 유의하세요.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EKIAUAXQT3SWDEXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/john",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "john"
    },
    "eventTime": "2021-07-10T19:56:13Z",
    "eventSource": "memorydb.amazonaws.com",
    "eventName": "CreateUser",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.01",
    "userAgent": "aws-cli/2.2.29 Python/3.9.6 Darwin/19.6.0 source/x86_64 prompt/off
 command/memorydb.create-user",
    "requestParameters": {
        "userName": "memorydb-user",
        "authenticationMode": {
            "type": "password",
            "passwords": [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ]
        },
        "accessString": "~* &* -@all +@read"
    },
    "responseElements": {
        "user": {
            "name": "memorydb-user",
```

```
            "status": "active",
            "accessString": "off ~* &* -@all +@read",
            "aCLNames": [],
            "minimumEngineVersion": "6.2",
            "authentication": {
                "type": "password",
                "passwordCount": 1
            },
            "aRN": "arn:aws:memorydb:us-east-1:123456789012:user/memorydb-user"
        }
    },
    "requestID": "ae288b5e-80ab-4ff8-989a-5ee5c67cd193",
    "eventID": "ed096e3e-16f1-4a23-866c-0baa6ec769f6",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}
```


MemoryDB의 규정 준수 확인

외부 감사자는 여러 AWS 규정 준수 프로그램의 일환으로 MemoryDB의 보안 및 규정 준수를 평가합니다. 여기에는 다음이 포함됩니다.

- PCI DSS(지불 카드 산업 데이터 보안 표준) 자세한 내용은 PCI DSS를 참조하세요.
- HIPAA(미국 건강 보험 양도 및 책임에 관한 법) BAA(비즈니스 제휴 계약) 자세한 내용은 HIPAA 규정 준수를 참조하세요.
- SOC(시스템 및 조직 제어) 1, 2, 3 자세한 내용은 SOC를 참조하세요.
- 연방정부의 위험 및 인증 관리 프로그램(FedRAMP) 보통. 자세한 내용은 FedRAMP 단원을 참조하세요.
- ISO/IEC 27001:2013, 27017:2015, 27018:2019, 및 ISO/IEC 9001:2015. 자세한 내용은 AWS ISO 및 CSA STAR 인증 및 서비스를 참조하세요.

특정 규정 준수 프로그램 범위에 속하는 AWS 서비스의 목록은 규정 준수 프로그램 제공 AWS 범위 내 서비스 페이지에서 확인하세요.

AWS Artifact(을)를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 AWS Artifact에서 보고서 다운로드를 참조하세요.

MemoryDB 사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 결정됩니다. AWS에서는 규정 준수를 지원할 다음과 같은 리소스를 제공합니다.

- 보안 및 규정 준수 빠른 시작 안내서: 이 배포 안내서에서는 아키텍처 고려 사항에 관해 설명하고 AWS에서 보안 및 규정 준수에 중점을 둔 기본 환경을 배포하기 위한 단계를 제공합니다.
- AWS 규정 준수 리소스: 고객 조직이 속한 산업 및 위치에 적용될 수 있는 워크북 및 가이드 콜렉션입니다.
- AWS Config개발자 가이드의 규칙을 사용하여 리소스 평가 – AWS Config을(를) 사용하여 리소스 구성이 내부 사례, 업계 지침, 규정을 얼마나 잘 준수하는지 평가합니다.
- AWS Security Hub CSPM: 이 AWS 서비스는 보안 산업 표준 및 모범 사례 규정 준수 여부를 확인하는 데 도움이 되도록 AWS 내 보안 상태를 종합적으로 보여줍니다.
- AWS Audit Manager – 이 AWS 서비스는 AWS 사용량을 지속적으로 감사하여 위험과 규정 및 산업 표준의 준수를 관리하는 방법을 간소화하는 데 도움이 됩니다.

MemoryDB의 인프라 보안

관리형 서비스인 MemoryDB는 Amazon Web Services: 보안 프로세스 개요 백서에 설명된 AWS 글로벌 네트워크 보안 절차로 보호됩니다.

AWS에서 게시한 API 직접 호출을 사용하여 네트워크를 통해 MemoryDB에 액세스합니다. 클라이언트가 전송 계층 보안(TLS) 1.2 이상을 지원해야 합니다. TLS 1.3 이상을 권장합니다. 클라이언트는 Ephemeral Diffie-Hellman(DHE) 또는 Elliptic Curve Ephemeral Diffie-Hellman(ECDHE)과 같은 PFS(전달 완전 보안, Perfect Forward Secrecy)가 포함된 암호 제품군도 지원해야 합니다. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 주체와 관련된 보안 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 보안 자격 증명을 생성하여 요청에 서명할 수 있습니다.

인터네트워크 트래픽 개인 정보

MemoryDB는 다음 기술을 사용하여 데이터 보안을 유지하고 무단 액세스로부터 보호합니다.

- MemoryDB 및 Amazon VPC은 설치에 필요한 보안 그룹 유형을 설명합니다.
- MemoryDB API 및 인터페이스 VPC 엔드포인트(AWS PrivateLink)는 VPC와 MemoryDB API 엔드포인트 간에 프라이빗 연결을 설정할 수 있게 합니다.

- MemoryDB의 ID 및 액세스 관리은 사용자, 그룹 및 역할의 작업을 부여하고 제한하는 용도입니다.

MemoryDB 및 Amazon VPC

Amazon Virtual Private Cloud(Amazon VPC) 서비스는 기존 데이터 센터와 매우 유사한 가상 네트워크를 정의합니다. Amazon VPC로 Virtual Private Cloud(VPC)를 구성할 때 그 IP 주소 범위를 선택하고 서브넷을 생성하고 라우팅 테이블, 네트워크 게이트웨이 및 보안 설정을 구성할 수 있습니다. 또한 Amazon VPC 보안 그룹을 사용하여 가상 네트워크에 클러스터를 추가하고 클러스터에 대한 액세스 권한을 제어할 수 있습니다.

이 단원에서는 VPC의 MemoryDB 클러스터를 수동으로 구성하는 방법을 설명합니다. 이 정보는 MemoryDB 및 Amazon VPC가 연동되는 방식을 더 깊이 이해하고자 하는 사용자를 대상으로 합니다.

주제

- MemoryDB 및 VPC에 대한 이해
- Amazon VPC에 있는 MemoryDB 클러스터에 액세스하기 위한 액세스 패턴
- Virtual Private Cloud(VPC) 생성

MemoryDB 및 VPC에 대한 이해

MemoryDB는 Amazon VPC와 완벽하게 통합되어 있습니다. MemoryDB 사용자의 경우, 이는 다음을 의미합니다.

- MemoryDB는 항상 VPC에서 클러스터를 시작합니다.
- AWS을(를) 처음 사용하는 경우, 기본 VPC가 자동으로 생성됩니다.
- 기본 VPC가 있고 클러스터를 시작할 때 서브넷을 지정하지 않으면 클러스터가 기본 Amazon VPC에서 시작합니다.

자세한 정보는 [Detecting Your Supported Platforms and Whether You Have a Default VPC]를 참조하세요.

Amazon VPC를 사용하면 기존의 데이터 센터와 매우 유사한 AWS 클라우드에 가상 네트워크를 생성할 수 있습니다. IP 주소 범위 선택, 서브넷 생성 및 라우팅 테이블, 네트워크 게이트웨이, 보안 설정 구성을 포함하여 VPC를 구성할 수 있습니다.

MemoryDB는 소프트웨어 업그레이드, 패치, 장애 감지 및 복구를 관리합니다.

VPC의 MemoryDB 개요

- VPC는 자체 IP 주소 블록이 할당된 AWS 클라우드의 격리된 부분입니다.
- 인터넷 게이트웨이는 VPC를 인터넷에 직접 연결하고 VPC 외부에서 실행되는 Amazon Simple Storage Service(Amazon S3)와 같은 다른 AWS 리소스에 대한 액세스를 제공합니다.
- Amazon VPC 서브넷은 보안 및 운영상의 필요에 따라 AWS 리소스를 격리할 수 있는 VPC의 IP 주소 범위 세그먼트입니다.
- Amazon VPC 보안 그룹은 MemoryDB 클러스터와 Amazon EC2 인스턴스의 인바운드 및 아웃바운드 트래픽을 제어합니다.
- 서브넷에서 MemoryDB 클러스터를 실행할 수 있습니다. 노드는 서브넷 주소 범위의 프라이빗 IP 주소를 가집니다.
- 또한 서브넷에서 Amazon EC2 인스턴스를 시작할 수 있습니다. 각각의 Amazon EC2 인스턴스는 서브넷 주소 범위의 프라이빗 IP 주소를 가집니다. Amazon EC2 인스턴스를 동일한 서브넷의 모든 노드에 연결할 수 있습니다.
- VPC의 Amazon EC2 인스턴스를 인터넷에 연결하려면 인스턴스에 탄력적 IP 주소라는 고정 퍼블릭 주소를 할당해야 합니다.

사전 조건

VPC에 MemoryDB 클러스터를 생성하려면 VPC가 다음 요구 사항을 충족해야 합니다.

- VPC는 비전용 Amazon EC2 인스턴스를 허용해야 합니다. 전용 인스턴스 테넌시로 구성된 VPC에서는 MemoryDB를 사용할 수 없습니다.
- VPC에 대해 서브넷 그룹을 정의해야 합니다. MemoryDB는 해당 서브넷 그룹을 사용하여 노드에 연결된 서브넷 내의 서브넷 및 IP 주소를 선택합니다.
- VPC에 대해 보안 그룹을 정의하거나 제공된 기본값을 사용할 수 있습니다.
- 각 서브넷의 CIDR 블록은 유지 관리 작업에서 MemoryDB에 사용할 여분의 IP 주소를 제공할 수 있을 만큼 충분히 커야 합니다.

라우팅 및 보안

VPC에서 라우팅을 구성하여 트래픽 흐름(예: 인터넷 게이트웨이, 가상 프라이빗 게이트웨이)을 제어할 수 있습니다. 인터넷 게이트웨이를 통해 VPC가 VPC에서 실행되지 않는 다른 AWS 리소스에 직접 액세스할 수 있습니다. 조직의 로컬 네트워크에 연결된 가상 사설 게이트웨이만을 사용하도록 선택한 경우, VPN을 통해 인터넷 바운드 트래픽을 라우팅하고 출구를 제어하기 위한 로컬 보안 정책 및 방화벽을 사용할 수 있습니다. 이 경우, 인터넷을 통해 AWS 리소스에 액세스할 때 대역폭 요금이 추가로 부과됩니다.

Amazon VPC 보안 그룹을 사용하여 Amazon VPC에서 MemoryDB 클러스터 및 Amazon EC2 인스턴스를 보호할 수 있습니다. 보안 그룹은 서브넷 레벨이 아닌 인스턴스 레벨에서 방화벽처럼 작동합니다.

> **Note**
>
> 기본 IP 주소가 시간이 지남에 따라 변경될 수 있으므로 노드에 연결할 때 DNS 이름을 사용하는 것이 좋습니다.

Amazon VPC 설명서

Amazon VPC에는 Amazon VPC를 생성하고 사용하는 방법을 설명하는 자체 문서 세트가 있습니다. 다음 테이블은 Amazon VPC 지침에서 정보를 찾을 수 있는 위치를 보여줍니다.

설명	설명서
Amazon VPC 사용을 시작하는 방법	Amazon VPC 시작하기

설명	설명서
AWS Management 콘솔을 통해 Amazon VPC 를 사용하는 방법	Amazon VPC User Guide
모든 Amazon VPC 명령의 전체 설명	Amazon EC2 명령줄 레퍼런스(Amazon VPC 명령은 Amazon EC2 참조에서 찾을 수 있음)
Amazon VPC API 작업, 데이터 형식 및 오류의 전체 설명	Amazon EC2 API 참조(Amazon VPC API 작업은 Amazon EC2 참조에서 찾을 수 있음)
선택적인 IPsec VPN 연결 사용자 측의 게이트웨이를 구성하는 데 필요한 네트워크 관리자를 위한 정보	AWS Site-to-Site VPN이란 무엇입니까?

Amazon Virtual Private Cloud에 대한 자세한 내용은 Amazon Virtual Private Cloud 섹션을 참조하세요.

Amazon VPC에 있는 MemoryDB 클러스터에 액세스하기 위한 액세스 패턴

MemoryDB에서는 Amazon VPC에 있는 클러스터에 액세스할 수 있도록 다음 시나리오를 지원합니다.

목차

MemoryDB 클러스터와 Amazon EC2 인스턴스가 같은 Amazon VPC에 있는 경우, MemoryDB 클러스터 액세스

가장 일반적인 사용 사례는 EC2 인스턴스에 배포된 애플리케이션이 같은 VPC에 있는 클러스터에 연결해야 하는 경우입니다.

동일한 VPC에서 EC2 인스턴스와 클러스터 간 액세스를 관리하는 가장 간단한 방법은 다음과 같습니다.

1. 클러스터의 VPC 보안 그룹을 만듭니다. 이 보안 그룹을 사용해 클러스터에 대한 액세스를 제한할 수 있습니다. 예를 들어, 클러스터를 만들 때 할당한 포트와 클러스터에 액세스할 때 이용할 IP 주소를 사용해 TCP 액세스를 허용하는 이 보안 그룹의 사용자 지정 규칙을 만들 수 있습니다.

   MemoryDB 클러스터의 기본 포트는 6379입니다.

2. EC2 인스턴스(웹 및 애플리케이션 서버)의 VPC 보안 그룹을 만듭니다. 이 보안 그룹은 필요할 경우 VPC의 라우팅 테이블을 통한 EC2 인스턴스 액세스를 허용할 수 있습니다. 예를 들어, 이 보안 그룹에서 TCP가 포트 22를 통해 EC2 인스턴스에 액세스하도록 허용하는 규칙을 설정할 수 있습니다.
3. 클러스터에 대한 보안 그룹에서 EC2 인스턴스에 대해 생성한 보안 그룹으로부터의 연결을 허용하는 사용자 지정 규칙을 만듭니다. 그러면 보안 그룹의 모든 구성원이 클러스터에 액세스하도록 허용됩니다.

VPC 보안 그룹에서 다른 보안 그룹으로부터의 연결을 허용하는 규칙을 만들려면

1. AWS 관리 콘솔에 로그인한 다음 https://console.aws.amazon.com/vpc에서 Amazon VPC 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 [Security Groups]를 선택합니다.
3. 클러스터에 사용할 보안 그룹을 선택하거나 만듭니다. [Inbound Rules]에서 [Edit Inbound Rules]를을 선택한 다음 [Add Rule][을 선택합니다. 이 보안 그룹은 다른 보안 그룹 멤버에 대한 액세스를 허용합니다.
4. [Type]에서 [Custom TCP Rule]을 선택합니다.

    a. [Port Range]에 대해 클러스터를 만들 때 사용한 포트를 지정합니다.

       MemoryDB 클러스터의 기본 포트는 6379입니다.

    b. [Source] 상자에 보안 그룹 ID를 입력합니다. 목록에서 Amazon EC2 인스턴스에 사용할 보안 그룹을 선택합니다.
5. 완료되면 [Save]를 선택합니다.

MemoryDB 클러스터와 Amazon EC2 인스턴스가 다른 Amazon VPC에 있는 경우, ElastiCache 클러스터 액세스

클러스터가 액세스에 사용할 EC2 인스턴스와 다른 VPC에 있는 경우, 여러 가지 방법으로 클러스터에 액세스할 수 있습니다. 클러스터와 EC2 인스턴스가 서로 다른 VPC에 있지만 리전은 동일한 경우, VPC 피어링을 사용할 수 있습니다. 클러스터와 EC2 인스턴스가 서로 다른 리전에 있으면 리전 간에 VPN 연결을 만들 수 있습니다.

주제

- ElastiCache 클러스터와 Amazon EC2 인스턴스가 같은 리전의 서로 다른 Amazon VPC에 있는 경우, MemoryDB 클러스터 액세스

- MemoryDB 클러스터와 Amazon EC2 인스턴스가 다른 리전의 다른 Amazon VPC에 있는 경우, MemoryDB 클러스터 액세스

ElastiCache 클러스터와 Amazon EC2 인스턴스가 같은 리전의 서로 다른 Amazon VPC에 있는 경우, MemoryDB 클러스터 액세스

같은 리전 내의 서로 다른 Amazon VPC에 있는 Amazon EC2 인스턴스가 액세스하는 클러스터 - VPC 피어링 연결

VPC 피어링 연결은 프라이빗 IP 주소를 사용하여 두 VPC 간에 트래픽을 라우팅할 수 있도록 하기 위한 두 VPC 사이의 네트워킹 연결입니다. 동일한 네트워크에 속하는 경우와 같이 VPC의 인스턴스가 서로 통신할 수 있습니다. 자체 Amazon VPC 간의 VPC 피어링 연결, 또는 단일 리전 내에 있는 다른 AWS 계정에서 Amazon VPC와의 VPC 피어링 연결을 생성할 수 있습니다. Amazon VPC 피어링에 대한 자세한 내용은 VPC 설명서를 참조하세요.

피어링으로 다른 Amazon VPC에 있는 클러스터에 액세스하려면

1. 두 VPC에 겹치는 IP 범위가 없거나 이 VPC를 피어링할 수 없어야 합니다.
2. 두 VPC를 피어링합니다. 자세한 정보는 Amazon VPC 피어링 연결 생성 및 수락을 참조하세요.
3. 라우팅 테이블을 업데이트합니다. 자세한 내용은 VPC 피어링 연결을 위한 라우팅 테이블 업데이트를 참조하세요.
4. MemoryDB 클러스터의 보안 그룹을 수정하여 피어링된 VPC의 애플리케이션 보안 그룹에서 들어오는 인바운드 연결을 허용합니다. 자세한 내용은 피어 VPC 보안 그룹 참조를 참조하세요.

피어링 연결을 통해 클러스터에 액세스하면 데이터 전송 비용이 추가로 발생합니다.

Transit Gateway 사용

Transit Gateway를 사용하면 VPC와 VPN 연결을 동일한 AWS 리전에 연결하고 두 리전 간의 트래픽을 라우팅할 수 있습니다. Transit Gateway는 AWS 계정에서 작동하며, AWS Resource Access Manager를 사용하여 Transit Gateway를 다른 계정과 공유할 수 있습니다. Transit Gateway를 다른 AWS 계정과 공유하면 계정 소유자가 자신의 VPC를 Transit Gateway에 연결할 수 있습니다. 두 계정의 사용자는 언제든지 연결을 삭제할 수 있습니다.

Transit Gateway에서 멀티캐스트를 활성화한 다음 도메인과 연결된 VPC 연결을 통해 멀티캐스트 소스에서 멀티캐스트 그룹 멤버로 멀티캐스트 트래픽을 보낼 수 있는 Transit Gateway 멀티캐스트 도메인을 생성할 수 있습니다.

또한 서로 다른 AWS 리전에 있는 Transit Gateway 간에 피어링 연결을 생성할 수도 있습니다. 이렇게 하면 여러 리전의 Transit Gateway 연결 간에 트래픽을 라우팅할 수 있습니다.

자세한 내용은 [전송 게이트웨이](#)를 참조하세요.

MemoryDB 클러스터와 Amazon EC2 인스턴스가 다른 리전의 다른 Amazon VPC에 있는 경우, MemoryDB 클러스터 액세스

전송 VPC 사용

VPC 피어링을 사용하는 대신, 지리적으로 떨어져 있는 여러 VPC와 원격 네트워크를 연결하는 또 다른 일반적인 전략은 글로벌 네트워크 전송 센터 역할을 하는 전송 VPC를 만드는 것입니다. 전송 VPC는 네트워크 관리를 간소화하고 여러 VPC와 원격 네트워크를 연결하는 데 필요한 연결 수를 최소화합니다. 이 디자인은 기존의 방식대로 공동 배치 전송 허브에 실제 존재를 만들거나 물리적 네트워크 장비를 배포하지 않고 가상으로 구현되므로 시간, 노력, 비용을 아낄 수 있습니다.

다른 리전의 다른 VPC를 지나는 연결

전송 Amazon VPC가 설정되면 한 리전의 “스포크” VPC에 배포된 애플리케이션이 다른 리전의 “스포크” VPC에 있는 MemoryDB 클러스터에 연결할 수 있습니다.

다른 AWS 리전의 다른 VPC에 있는 클러스터에 액세스하려면

1. 전송 VPC 솔루션을 배포합니다. 자세한 내용은 [AWS Transit Gateway](#)를 참조하세요.
2. 앱 및 VPC에서 VPC 라우팅 테이블을 업데이트하여 VGW(가상 프라이빗 게이트웨이) 및 VPN 어플라이언스를 통해 트래픽을 라우팅합니다. BGP(Border Gateway Protocol)를 사용하는 동적 라우팅의 경우 라우팅이 자동으로 전파됩니다.
3. MemoryDB 클러스터의 보안 그룹을 수정하여 애플리케이션 인스턴스 IP 범위에서 들어오는 인바운드 연결을 허용합니다. 이 시나리오에는 애플리케이션 서버 보안 그룹을 참조할 수 없습니다.

여러 리전의 클러스터에 액세스하면 네트워크 지연 시간이 생기고 교차 리전 데이터 전송 비용이 추가로 발생합니다.

고객의 데이터 센터에서 실행되는 애플리케이션에서 MemoryDB 클러스터 액세스

고객 데이터 센터의 클라이언트나 애플리케이션이 VPC의 MemoryDB 클러스터에 액세스해야 하는 하이브리드 아키텍처도 가능한 시나리오입니다. VPN이나 Direct Connect를 통해 고객의 VPC와 데이터 센터 간에 연결을 제공하여 이 시나리오도 지원됩니다.

주제

- VPN 연결을 사용하여 고객의 데이터 센터에서 실행되는 애플리케이션에서 MemoryDB 클러스터 액세스
- Direct Connect를 사용하여 고객의 데이터 센터에서 실행되는 애플리케이션에서 MemoryDB 클러스터 액세스

VPN 연결을 사용하여 고객의 데이터 센터에서 실행되는 애플리케이션에서 MemoryDB 클러스터 액세스

VPN을 통해 데이터 센터에서 MemoryDB에 연결

VPN 연결을 통해 온-프레미스 애플리케이션에서 VPC의 클러스터에 액세스하려면

1. 하드웨어 가상 프라이빗 게이트웨이를 VPC에 추가하여 VPN 연결을 설정합니다. 자세한 내용은 VPC에 하드웨어 가상 프라이빗 게이트웨이 추가를 참조하세요.
2. MemoryDB 클러스터가 배포되는 서브넷의 VPC 라우팅 테이블을 업데이트하여 온프레미스 애플리케이션 서버에서 들어오는 트래픽을 허용합니다. BGP를 사용하는 동적 라우팅의 경우 라우팅이 자동으로 전파될 수 있습니다.
3. MemoryDB 클러스터의 보안 그룹을 수정하여 온프레미스 애플리케이션 서버에서 들어오는 인바운드 연결을 허용합니다.

VPN 연결을 통해 클러스터에 액세스하면 네트워크 지연 시간이 생기고 데이터 전송 비용이 추가로 발생합니다.

Direct Connect를 사용하여 고객의 데이터 센터에서 실행되는 애플리케이션에서 MemoryDB 클러스터 액세스

Direct Connect를 통해 데이터 센터에서 MemoryDB에 연결

Direct Connect를 사용하여 네트워크에서 실행되는 애플리케이션에서 MemoryDB 클러스터에 액세스하려면

1. Direct Connect 연결을 설정합니다. 자세한 내용은 AWS Direct Connect 시작하기를 참조하세요.
2. MemoryDB 클러스터의 보안 그룹을 수정하여 온프레미스 애플리케이션 서버에서 들어오는 인바운드 연결을 허용합니다.

DX 연결을 통해 클러스터에 액세스하면 네트워크 지연 시간이 생기고 데이터 전송 요금이 추가로 발생할 수 있습니다.

Virtual Private Cloud(VPC) 생성

이 예제에서는 각 가용 영역에 대해 프라이빗 서브넷이 있는 Amazon VPC서비스를 기반으로 Virtual Private Cloud(VPC)를 생성합니다.

VPC 생성(콘솔)

Amazon Virtual Private Cloud 내에서 MemoryDB 클러스터를 생성하려면

1. AWS 관리 콘솔에 로그인한 다음 https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
2. VPC 대시보드에서 VPC 생성(Create VPC)을 선택합니다.
3. 생성할 리소스(Resources to create)에서 VPC 등(VPC and more)을 선택합니다.
4. 가용 영역(AZ) 수(Number of Availability Zones(AZs))에서 서브넷을 시작할 가용 영역 수를 선택합니다.
5. 퍼블릭 서브넷 수(Number of public subnets)에서 VPC 추가할 퍼블릭 서브넷 수를 선택합니다.
6. 프라이빗 서브넷 수(Number of private subnets)에서 VPC 추가할 프라이빗 서브넷 수를 선택합니다.

   > **Tip**
   >
   > 서브넷 식별자(퍼블릭 서브넷, 프라이빗 서브넷)를 기록해 둡니다. 추후 클러스터를 시작하고 Amazon VPC에 Amazon EC2 인스턴스를 추가하는 경우 이 정보가 필요합니다.

7. Amazon VPC 보안 그룹을 생성합니다. 클러스터 및 Amazon EC2 인스턴스에 이 그룹을 사용합니다.
   a. 왼쪽 탐색 창에서 AWS Management 콘솔, Security Groups를 선택합니다.
   b. 보안 그룹 생성을 선택합니다.
   c. 보안 그룹의 이름과 설명을 해당 상자에 입력합니다. VPC의 경우, VPC의 식별자를 선택합니다.
   d. 원하는 대로 설정되었으면 [Yes, Create]를 선택합니다.
8. 보안 그룹의 네트워크 수신 규칙을 정의합니다. 이 규칙을 사용하면 SSH(Secure Shell)를 사용하여 Amazon EC2 인스턴스에 연결할 수 있습니다.
   a. 왼쪽 탐색 창에서 [Security Groups]를 선택합니다.
   b. 목록에서 보안 그룹을 찾아 선택합니다.

c. [Security Group] 아래에서 [Inbound] 탭을 선택합니다. [Create a new rule] 상자에서 [SSH]를 선택한 다음 [Add Rule]을 선택합니다.

   새 인바운드 규칙에 다음 값을 설정하여 HTTP 액세스를 허용합니다.

   - 유형: HTTP
   - 소스: 0.0.0.0/0

d. 새 인바운드 규칙에 다음 값을 설정하여 HTTP 액세스를 허용합니다.

   - 유형: HTTP
   - 소스: 0.0.0.0/0

   [Apply Rule Changes]를 선택합니다.

이제 서브넷 그룹을 생성하고 Amazon VPC에서 클러스터를 생성할 수 있습니다.

서브넷 및 서브넷 그룹

서브넷 그룹은 Amazon Virtual Private Cloud(VPC) 환경에서 실행 중인 클러스터에 대해 지정할 수 있는 서브넷(일반적으로 프라이빗 서브넷) 모음입니다.

Amazon VPC에서 클러스터를 생성하는 경우 서브넷 그룹을 지정하거나 제공된 기본 서브넷 그룹을 사용할 수 있습니다. MemoryDB는 해당 서브넷 그룹을 사용하여 노드에 연결된 서브넷 내의 서브넷 및 IP 주소를 선택합니다.

이 섹션에서는 서브넷과 서브넷 그룹을 생성 및 활용하여 MemoryDB 리소스에 대한 액세스를 관리하는 방법을 알아봅니다.

Amazon VPC 환경에서 서브넷 그룹 사용에 대한 자세한 내용은 3단계: 클러스터에 대한 액세스 허가 섹션을 참조하세요.

지원되는 MemoryDB AZ ID

리전 이름/리전	지원되는 AZ ID		
미국 동부(오하이오) 리전 `us-east-2`	`use2-az1, use2-az2, use2-az3`		

리전 이름/리전	지원되는 AZ ID		
미국 동부(버지니아 북부) 리전 us-east-1	use1-az1, use1-az2, use1-az4, use1-az5, use1-az6		
미국 서부(캘리포니아 북부) 리전 us-west-1	usw1-az1, usw1-az2, usw1-az3		
미국 서부(오리건) 리전 us-west-2	usw2-az1, usw2-az2, usw2-az3, usw2-az4		
캐나다(중부) 리전 ca-central-1	cac1-az1, cac1-az2, cac1-az4		
아시아 태평양(홍콩) 리전 ap-east-1	ape1-az1, ape1-az2, ape1-az3		
아시아 태평양(뭄바이) 리전 ap-south-1	aps1-az1, aps1-az2, aps1-az3		
아시아 태평양(도쿄) 리전 ap-northeast-1	apne1-az1, apne1-az2, apne1-az4		
아시아 태평양(서울) 리전 ap-northeast-2	apne2-az1, apne2-az2, apne2-az3		

리전 이름/리전	지원되는 AZ ID		
아시아 태평양(싱가포르) 리전 `ap-southeast-1`	`apse1-az1, apse1-az2, apse1-az3`		
아시아 태평양(시드니) 리전 `ap-southeast-2`	`apse2-az1, apse2-az2, apse2-az3`		
유럽(프랑크푸르트) 리전 `eu-central-1`	`euc1-az1, euc1-az2, euc1-az3`		
유럽(아일랜드) 리전 `eu-west-1`	`euw1-az1, euw1-az2, euw1-az3`		
유럽(런던) 리전 `eu-west-2`	`euw2-az1, euw2-az2, euw2-az3`		
EU(파리) 리전 `eu-west-3`	`euw3-az1, euw3-az2, euw3-az3`		
유럽(스톡홀름) 리전 `eu-north-1`	`eun1-az1, eun1-az2, eun1-az3`		
유럽(밀라노) 리전 `eu-south-1`	`eus1-az1, eus1-az2, eus1-az3`		
남아메리카(상파울루) 리전 `sa-east-1`	`sae1-az1, sae1-az2, sae1-az3`		

리전 이름/리전	지원되는 AZ ID		
중국(베이징) 리전 cn-north-1	cnn1-az1, cnn1-az2		
중국(닝샤) 리전 cn-northwest-1	cnw1-az1, cnw1-az2, cnw1-az3		
us-gov-east-1	usge1-az1, usge1-az2, usge1-az3		
us-gov-west-1	usgw1-az1, usgw1-az2, usgw1-az3		
유럽(스페인) 리전 eu-south-2	eus2-az1, eus2-az2, eus2-az3		

주제

- MemoryDB 및 IPV6
- 서브넷 그룹 생성
- 서브넷 그룹 업데이트
- 서브넷 그룹 세부 정보 보기
- 서브넷 그룹 삭제

MemoryDB 및 IPV6

듀얼 스택 및 ipv6 전용 서브넷이 있는 서브넷 그룹을 제공하여 Valkey 및 Redis OSS 엔진으로 새 듀얼 스택 및 ipv6 전용 클러스터를 생성할 수 있습니다. 기존 클러스터의 네트워크 유형을 변경할 수 없습니다.

이 기능을 사용하면 다음을 수행할 수 있습니다.

- 듀얼 스택 서브넷에서 ipv4 전용 및 듀얼 스택 클러스터를 생성합니다.
- ipv6 전용 서브넷에서 ipv6 전용 클러스터를 생성합니다.
- ipv4 전용, 듀얼 스택 및 ipv6 전용 서브넷을 지원하는 새 서브넷 그룹을 생성합니다.
- 기본 VPC의 추가 서브넷을 포함하도록 기존 서브넷 그룹을 수정합니다.
- 서브넷 그룹의 기존 서브넷 수정
  - IPv6 전용 서브넷을 IPv6용으로 구성된 서브넷 그룹에 추가
  - IPv4 및 듀얼 스택 지원을 위해 구성된 서브넷 그룹에 IPv4 또는 듀얼 스택 서브넷 추가
- 듀얼 스택 및 ipv6 클러스터에 대한 엔진 검색 명령을 통해 ipv4 또는 ipv6 주소가 있는 클러스터의 모든 노드를 검색합니다. 이러한 검색 명령에는 `redis_info`, `redis_cluster`등이 포함됩니다.
- 듀얼 스택 및 ipv6 클러스터에 대한 DNS 검색 명령을 통해 클러스터에 있는 모든 노드의 ipv4 및 ipv6 주소를 검색합니다.

서브넷 그룹 생성

새 서브넷 그룹을 생성할 때 사용 가능한 IP 주소의 수를 기록하세요. 서브넷에 무료 IP 주소가 거의 없는 경우 클러스터에 추가할 수 있는 노드의 수가 제약될 수 있습니다. 이 문제를 해결하기 위해 클러스터의 가용 영역에 충분한 수의 IP 주소가 있도록 서브넷 그룹에 하나 이상의 서브넷을 지정할 수 있습니다. 그 이후 클러스터에 더 많은 노드를 추가할 수 있습니다.

다음 절차는 `mysubnetgroup`(콘솔)이라는 서브넷 그룹, AWS CLI, MemoryDB API를 생성하는 방법을 보여 줍니다.

서브넷 그룹 생성(콘솔)

다음 절차는 서브넷 그룹을 생성하는 방법을 보여줍니다(콘솔).

서브넷 그룹을 생성하는 방법(콘솔)

1. AWS 관리 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.

2. 왼쪽 탐색 창에서 서브넷 그룹을 선택합니다.
3. Create Subnet Group을 선택합니다.
4. Create Subnet Group 창에서 다음을 수행하세요.

    a. Name 상자에 서브넷 그룹의 이름을 입력합니다.

        클러스터 명명 제약 조건은 다음과 같습니다.

        - 1~40자의 영숫자 또는 하이픈으로 구성되어야 합니다.
        - 문자로 시작해야 합니다.
        - 하이픈 2개가 연속될 수 없습니다.
        - 끝에 하이픈이 올 수 없습니다.

    b. Description 상자에 서브넷 그룹에 대한 설명을 입력합니다.

    c. VPC ID 상자에서 생성한 Amazon VPC를 선택합니다. 아직 만들지 않은 경우, VPC 생성 버튼을 선택하고 단계에 따라 VPC를 생성하세요.

    d. 선택한 서브넷에서 가용 영역과 프라이빗 서브넷의 ID를 선택한 다음 선택을 선택합니다.

5. 태그의 경우 선택적으로 태그를 적용하여 서브넷을 검색 및 필터링하거나 AWS 비용을 추적할 수 있습니다.
6. 원하는 대로 모두 설정되었으면 Create를 선택합니다.
7. 나타나는 확인 메시지에서 Close를 선택합니다.

새 DB 서브넷 그룹이 MemoryDB 콘솔의 서브넷 그룹 목록에 나타납니다. 창 하단에서 서브넷 그룹을 선택하여 이 그룹과 연결된 모든 서브넷 등의 상세 내용을 확인할 수 있습니다.

서브넷 그룹 생성(AWS CLI)

명령 프롬프트에서 `create-subnet-group` 명령을 사용하여 서브넷 그룹을 생성합니다.

Linux, macOS, Unix의 경우:

```
aws memorydb create-subnet-group \
    --subnet-group-name mysubnetgroup \
    --description "Testing" \
    --subnet-ids subnet-53df9c3a
```

Windows의 경우:

```
aws memorydb create-subnet-group ^
    --subnet-group-name mysubnetgroup ^
    --description "Testing" ^
    --subnet-ids subnet-53df9c3a
```

이 명령은 다음과 유사한 출력을 생성합니다.

```
    {
        "SubnetGroup": {
            "Subnets": [
                {
                    "Identifier": "subnet-53df9c3a",
                    "AvailabilityZone": {
                    "Name": "us-east-1a"
                    }
                }
            ],
            "VpcId": "vpc-3cfaef47",
            "Name": "mysubnetgroup",
            "ARN": "arn:aws:memorydb:us-east-1:012345678912:subnetgroup/
mysubnetgroup",
            "Description": "Testing"
        }
    }
```

자세한 내용은 AWS CLI 항목 create-subnet-group를 참조하세요.

서브넷 그룹 생성(MemoryDB API)

MemoryDB API를 사용하여 다음 파라미터와 함께 `CreateSubnetGroup`을(를) 직접적으로 호출합니다.

- `SubnetGroupName=mysubnetgroup`
- `Description=Testing`
- `SubnetIds.member.1=subnet-53df9c3a`

서브넷 그룹 업데이트

서브넷 그룹의 설명을 업데이트하거나 서브넷 그룹과 연관된 서브넷 ID의 목록을 수정할 수 있습니다. 클러스터가 현재 해당 서브넷을 사용하고 있는 경우 서브넷 그룹에서 서브넷 ID를 삭제할 수 없습니다.

다음 절차는 서브넷 그룹을 업데이트하는 방법을 보여줍니다.

서브넷 그룹 업데이트(콘솔)

서브넷 그룹을 업데이트하는 방법

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 서브넷 그룹을 선택합니다.
3. 서브넷 그룹 목록에서 수정하려는 서브넷 그룹을 선택합니다.
4. 이름, VPCID 및 설명 필드는 수정할 수 없습니다.
5. 선택한 서브넷 섹션에서 관리를 클릭하여 서브넷에 필요한 가용 영역을 변경합니다. 변경 사항을 저장하려면 Save를 선택합니다.

서브넷 그룹 업데이트(AWS CLI)

명령 프롬프트에서 `update-subnet-group` 명령을 사용하여 서브넷 그룹을 업데이트합니다.

Linux, macOS, Unix의 경우:

```
aws memorydb update-subnet-group \
    --subnet-group-name mysubnetgroup \
    --description "New description" \
    --subnet-ids "subnet-42df9c3a" "subnet-48fc21a9"
```

Windows의 경우:

```
aws memorydb update-subnet-group ^
    --subnet-group-name mysubnetgroup ^
    --description "New description" ^
    --subnet-ids "subnet-42df9c3a" "subnet-48fc21a9"
```

이 명령은 다음과 유사한 출력을 생성합니다.

```
{
    "SubnetGroup": {
        "VpcId": "vpc-73cd3c17",
        "Description": "New description",
        "Subnets": [
            {
                "Identifier": "subnet-42dcf93a",
                "AvailabilityZone": {
                    "Name": "us-east-1a"
                }
            },
            {
                "Identifier": "subnet-48fc12a9",
                "AvailabilityZone": {
                    "Name": "us-east-1a"
                }
            }
        ],
        "Name": "mysubnetgroup",
        "ARN": "arn:aws:memorydb:us-east-1:012345678912:subnetgroup/mysubnetgroup",
    }
}
```

자세한 내용은 update-subnet-group AWS CLI 항목을 참조하세요.

서브넷 그룹 업데이트(MemoryDB API)

MemoryDB API를 사용하여 다음 파라미터와 함께 `UpdateSubnetGroup`을(를) 직접적으로 호출합니다.

- `SubnetGroupName=`*`mysubnetgroup`*
- 변경할 값이 있는 다른 파라미터. 이 예제는 `Description=`*`New%20description`*을 사용하여 서브넷 그룹의 설명을 변경합니다.

Example

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=UpdateSubnetGroup
    &Description=New%20description
    &SubnetGroupName=mysubnetgroup
    &SubnetIds.member.1=subnet-42df9c3a
```

```
    &SubnetIds.member.2=subnet-48fc21a9
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Timestamp=20141201T220302Z
    &Version=2014-12-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Credential=<credential>
    &X-Amz-Date=20141201T220302Z
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Signature=<signature>
    &X-Amz-SignedHeaders=Host
```

> **Note**
>
> 새 서브넷 그룹을 생성할 때 사용 가능한 IP 주소의 수를 기록하세요. 서브넷에 무료 IP 주소가 거의 없는 경우 클러스터에 추가할 수 있는 노드의 수가 제약될 수 있습니다. 이 문제를 해결하기 위해 클러스터의 가용 영역에 충분한 수의 IP 주소가 있도록 서브넷 그룹에 하나 이상의 서브넷을 지정할 수 있습니다. 그 이후 클러스터에 더 많은 노드를 추가할 수 있습니다.

서브넷 그룹 세부 정보 보기

다음 절차는 서브넷 그룹을 보는 방법을 보여줍니다.

서브넷 그룹 세부 정보 보기(콘솔)

서브넷 그룹 세부 정보를 보는 방법(콘솔)

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 서브넷 그룹을 선택합니다.
3. 서브넷 그룹 페이지의 이름에서 서브넷 그룹을 선택하거나 검색 창에 서브넷 그룹 이름을 입력합니다.
4. 서브넷 그룹 페이지의 이름에서 서브넷 그룹을 선택하거나 검색 창에 서브넷 그룹 이름을 입력합니다.
5. 서브넷 그룹 설정에서 서브넷 그룹의 이름, 설명, VPC ID 및 Amazon 리소스 이름(ARN)을 볼 수 있습니다.
6. 서브넷에서 서브넷 그룹의 가용 영역, 서브넷 ID 및 CIDR 블록을 볼 수 있습니다.
7. 태그에서 서브넷 그룹과 관련된 모든 태그를 볼 수 있습니다.

서브넷 그룹 세부 정보 보기(AWS CLI)

명령 프롬프트에서 `describe-subnet-groups` 명령을 사용하여 지정된 서브넷 그룹 세부 정보를 확인합니다.

Linux, macOS, Unix의 경우:

```
aws memorydb describe-subnet-groups \
    --subnet-group-name mysubnetgroup
```

Windows의 경우:

```
aws memorydb describe-subnet-groups ^
    --subnet-group-name mysubnetgroup
```

이 명령은 다음과 유사한 출력을 생성합니다.


```
{
  "subnetgroups": [
    {
      "Subnets": [
        {
          "Identifier": "subnet-060cae3464095de6e",
          "AvailabilityZone": {
            "Name": "us-east-1a"
          }
        },
        {
          "Identifier": "subnet-049d11d4aa78700c3",
          "AvailabilityZone": {
            "Name": "us-east-1c"
          }
        },
        {
          "Identifier": "subnet-0389d4c4157c1edb4",
          "AvailabilityZone": {
            "Name": "us-east-1d"
          }
        }
      ],
      "VpcId": "vpc-036a8150d4300bcf2",
      "Name": "mysubnetgroup",
```

```
      "ARN": "arn:aws:memorydb:us-east-1:53791xzzz7620:subnetgroup/mysubnetgroup",
      "Description": "test"
    }
  ]
}
```

모든 서브넷 그룹의 세부 정보를 보려면 서브넷 그룹 이름을 지정하지 않고 동일한 명령을 사용합니다.

```
aws memorydb describe-subnet-groups
```

자세한 내용은 AWS CLI 항목 describe-subnet-groups를 참조하세요.

서브넷 그룹 보기(MemoryDB API)

MemoryDB API를 사용하여 다음 파라미터와 함께 `DescribeSubnetGroups`을(를) 직접적으로 호출합니다.

`SubnetGroupName=`*`mysubnetgroup`*

Example

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=UpdateSubnetGroup
    &Description=New%20description
    &SubnetGroupName=mysubnetgroup
    &SubnetIds.member.1=subnet-42df9c3a
    &SubnetIds.member.2=subnet-48fc21a9
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Timestamp=20211801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Credential=<credential>
    &X-Amz-Date=20210801T220302Z
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Signature=<signature>
    &X-Amz-SignedHeaders=Host
```

서브넷 그룹 삭제

서브넷 그룹이 더 이상 필요하지 않다고 판단되면 삭제할 수 있습니다. 현재 클러스터에서 사용 중인 경우 서브넷 그룹을 삭제할 수 없습니다. 또한 다중 AZ가 활성화된 클러스터에서 서브넷 그룹을 삭제할 경우 해당 클러스터에 서브넷이 2개 미만으로 남는다면 삭제할 수 없습니다. 먼저 다중 AZ를 선택 취소한 다음 서브넷을 삭제해야 합니다.

다음 절차는 서브넷 그룹을 삭제하는 방법을 보여줍니다.

서브넷 그룹 삭제(콘솔)

서브넷 그룹을 삭제하는 방법

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 서브넷 그룹을 선택합니다.
3. 서브넷 그룹 목록에서 삭제할 서브넷 그룹을 선택한 다음 작업을 선택하고 삭제를 선택합니다.

   > **Note**
   >
   > 기본 서브넷 그룹이나 클러스터와 연결된 서브넷 그룹은 삭제할 수 없습니다.

4. Delete Parameter Groups 확인 화면이 나타납니다.
5. 서브넷 그룹을 삭제하려면 확인 텍스트 상자에 `delete`을 입력합니다. 서브넷 그룹을 유지하려면 취소를 선택합니다.

서브넷 그룹 삭제(AWS CLI)

AWS CLI를 사용하여 다음 파라미터로 delete-subnet-group 명령을 호출할 수 있습니다.

- `--subnet-group-name` *`mysubnetgroup`*

Linux, macOS, Unix의 경우:

```
aws memorydb delete-subnet-group \
    --subnet-group-name mysubnetgroup
```

Windows의 경우:

```
aws memorydb delete-subnet-group ^
    --subnet-group-name mysubnetgroup
```

자세한 내용은 AWS CLI 주제의 delete-subnet-group를 참조하세요.

서브넷 그룹 삭제(MemoryDB API)

MemoryDB API를 사용하여 다음 파라미터와 함께 `DeleteSubnetGroup`을 직접적으로 호출합니다.

- `SubnetGroupName=mysubnetgroup`

Example

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DeleteSubnetGroup
    &SubnetGroupName=mysubnetgroup
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Credential=<credential>
    &X-Amz-Date=20210801T220302Z
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Signature=<signature>
    &X-Amz-SignedHeaders=Host
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 MemoryDB API 주제 DeleteSubnetGroup을 참조하세요.

MemoryDB API 및 인터페이스 VPC 엔드포인트(AWS PrivateLink)

인터페이스 VPC 엔드포인트를 생성하여 VPC와 Amazon MemoryDB API 엔드포인트 간에 프라이빗 연결을 설정할 수 있습니다. 인터페이스 엔드포인트는 AWS PrivateLink에 의해 구동됩니다. AWS PrivateLink를 사용하면 인터넷 게이트웨이, NAT 디바이스, VPN 연결 또는 AWS Direct Connect 연결 없이도 MemoryDB API 작업에 프라이빗 액세스할 수 있습니다.

VPC에 있는 인스턴스는 퍼블릭 IP 주소가 없어도 MemoryDB API 엔드포인트와 통신할 수 있습니다. 또한 인스턴스가 사용 가능한 MemoryDB API 작업을 사용하기 위해 퍼블릭 IP 주소가 필요하지 않습

니다. VPC와 MemoryDB 간의 트래픽은 Amazon 네트워크를 벗어나지 않습니다. 각 인터페이스 엔드포인트는 서브넷에서 하나 이상의 탄력적 네트워크 인터페이스로 표현됩니다. 탄력적 네트워크 인터페이스에 대한 자세한 내용은 Amazon EC2 사용 설명서의 탄력적 네트워크 인터페이스를 참조하세요.

- VPC 엔드포인트에 대한 자세한 정보는 Amazon VPC 사용 설명서의 인터페이스 VPC 엔드포인트 (AWS PrivateLink)를 참조하세요.
- MemoryDB API 작업에 관한 자세한 내용은 MemoryDB API 작업을 참조하세요.

인터페이스 VPC 엔드포인트를 생성한 후 엔드포인트에 대해 프라이빗 DNS 호스트 이름을 활성화하는 경우 기본 MemoryDB 엔드포인트(https://memorydb.*Region*.amazonaws.com)는 VPC 엔드포인트로 귀결됩니다. 프라이빗 DNS 호스트 이름을 활성화하지 않은 경우, Amazon VPC는 다음 형식으로 사용할 수 있는 DNS 엔드포인트를 제공합니다.

```
VPC_Endpoint_ID.memorydb.Region.vpce.amazonaws.com
```

자세한 정보는 Amazon VPC 사용 설명서에서 인터페이스 VPC 엔드포인트(AWS PrivateLink)를 참조하세요. MemoryDB는 VPC 내부에 있는 모든 API 작업에 대한 직접적인 호출 수행을 지원합니다.

> **Note**
>
> 프라이빗 DNS 호스트 이름은 VPC에 있는 하나의 VPC 엔드포인트에 대해서만 사용 설정할 수 있습니다. 추가 VPC 엔드포인트를 생성하려는 경우 프라이빗 DNS 호스트 이름을 사용 중지해야 합니다.

VPC 엔드포인트 고려 사항

MemoryDB API 엔드포인트에 대한 인터페이스 VPC 엔드포인트를 설정하기 전에 Amazon VPC 사용 설명서에서 인터페이스 엔드포인트 속성 및 제한 사항을 검토해야 합니다. MemoryDB 리소스 관리와 관련된 모든 MemoryDB API 작업은 AWS PrivateLink를 사용하여 VPC에서 사용할 수 있습니다. VPC 엔드포인트 정책은 MemoryDB API 엔드포인트에 대해 지원됩니다. 기본적으로, 엔드포인트를 통해 MemoryDB API 작업에 대한 전체 액세스가 허용됩니다. 자세한 정보는 Amazon VPC 사용 설명서의 VPC 엔드포인트를 통해 서비스에 대한 액세스 제어를 참조하세요.

MemoryDB API에 대한 인터페이스 VPC 엔드포인트 생성

Amazon VPC 콘솔 또는 AWS CLI를 사용하여 MemoryDB API에 대한 VPC 엔드포인트를 생성할 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트 생성을 참조하세요.

인터페이스 VPC 엔드포인트를 생성한 후 엔드포인트에 대한 프라이빗 DNS 호스트 이름을 사용할 수 있습니다. 그러면 기본 MemoryDB 엔드포인트(https://memorydb.*Region*.amazonaws.com)가 VPC 엔드포인트로 확인됩니다. 자세한 내용은 Amazon VPC 사용 설명서의 [인터페이스 엔드포인트를 통해 서비스 액세스](https://docs.aws.amazon.com/vpc/latest/userguide/vpce-interface.html)를 참조하세요.

Amazon MemoryDB API에 대한 VPC 엔드포인트 정책 생성

MemoryDB API에 대한 액세스를 제어하는 VPC 엔드포인트에 엔드포인트 정책을 연결할 수 있습니다. 이 정책은 다음을 지정합니다.

- 작업을 수행할 수 있는 보안 주체.
- 수행할 수 있는 작업.
- 작업을 수행할 수 있는 리소스.

자세한 정보는 Amazon VPC 사용 설명서의 [VPC 엔드포인트를 통해 서비스에 대한 액세스 컨트롤을](https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints-access.html) 참조하세요.

Example MemoryDB API 작업에 대한 VPC 엔드포인트 정책

다음은 MemoryDB API에 대한 엔드포인트 정책의 예입니다. 이 정책은 엔드포인트에 연결될 때 모든 리소스의 모든 보안 주체에 대한 액세스 권한을 나열된 MemoryDB API 작업에 부여합니다.

```
{
 "Statement": [{
  "Principal": "*",
  "Effect": "Allow",
  "Action": [
   "memorydb:CreateCluster",
   "memorydb:UpdateCluster",
   "memorydb:CreateSnapshot"
  ],
  "Resource": "*"
 }]
}
```

Example 지정된 AWS 계정의 모든 액세스를 거부하는 VPC 엔드포인트 정책

다음 VPC 엔드포인트 정책은 AWS 계정 *123456789012*가 엔드포인트를 사용하는 리소스에 대한 모든 액세스를 거부합니다. 이 정책은 다른 계정의 모든 작업을 허용합니다.

```
{
```

```
  "Statement": [{
    "Action": "*",
    "Effect": "Allow",
    "Resource": "*",
    "Principal": "*"
   },
   {
    "Action": "*",
    "Effect": "Deny",
    "Resource": "*",
    "Principal": {
     "AWS": [
      "123456789012"
     ]
    }
   }
  ]
 }
```

일반적인 취약성 및 노출(CVE): MemoryDB에서 해결된 보안 취약성

일반적인 취약성 및 노출(CVE)은 공개적으로 알려진 사이버 보안 취약성에 대한 항목의 목록입니다. 각 항목은 ID 번호, 설명 및 하나 이상의 공개 참조를 포함하는 링크입니다. 이 페이지에서 MemoryDB에서 해결된 보안 취약성 목록을 찾을 수 있습니다.

알려진 취약성으로부터 보호하려면 항상 최신 MemoryDB 버전으로 업그레이드하는 것이 좋습니다. MemoryDB는 PATCH 구성 요소를 노출합니다. 패치 버전은 이전 버전과 호환되는 버그 수정, 보안 수정 및 비기능 변경용입니다.

다음 표를 사용하여 특정 버전의 MemoryDB에 특정 보안 취약성에 대한 수정 사항이 있는지 확인할 수 있습니다. MemoryDB 캐시가 서비스 업데이트 보류 중인 경우 아래 나열된 보안 취약성 중 하나에 취약할 수 있습니다. 모든 서비스 업데이트를 적용하는 것이 좋습니다. 지원되는 MemoryDB 엔진 버전과 업그레이드 방법에 대한 자세한 내용은 엔진 버전 섹션을 참조하세요.

> **Note**
>
> - CVE가 MemoryDB 버전에서 처리되는 경우 이는 최신 버전에서도 처리된다는 의미입니다.
> - 다음 표의 별표(*)는 보안 취약성을 해결하기 위해 지정된 버전을 실행하는 MemoryDB 클러스터에 최신 서비스 업데이트가 적용되어야 함을 나타냅니다. 클러스터가 실행 중인

MemoryDB 버전에 최신 서비스 업데이트가 적용되었는지 확인하는 방법에 대한 자세한 내용은 서비스 업데이트 관리 섹션을 참조하세요.

MemoryDB 버전	해결된 CVEs
Valkey 7.3 및 Valkey의 모든 이전 버전 Redis OSS 7.1 및 Redis OSS의 모든 이전 버전	CVE-2025-49844*, CVE-2025-46817*, CVE-2025-46818*, CVE-2025-46819*
Valkey 7.2 및 7.3	CVE-2025-21607*, CVE-2025-21605*, CVE-2024-31449*, CVE-2024-31227*, CVE-2024-31228*
Valkey 7.2.7	CVE-2024-51741
Redis OSS 7.1 및 6.2	CVE-2025-21605*, CVE-2024-31449*, CVE-2024-31227*, CVE-2024-31228*, CVE-2023-41056
Redis OSS 7.0.7	CVE-2023-41056*
Redis OSS 6.2.7	CVE-2024-46981
Redis OSS 6.2.6	CVE-2022-24834*, CVE-2022-35977*, CVE-2022-36021*, CVE-2023-22458, CVE-2023-25155, CVE-2023-28856 CVE-2023-45145:이 CVE는 Redis OSS 6.2 및 7.0에서는 해결되었지만 Redis OSS 7.1에서는 해결되지 않았습니다.
Redis OSS 6.0.5	CVE-2022-24735*, CVE-2022-24736*

MemoryDB의 서비스 업데이트

MemoryDB는 클러스터 및 노드 플릿을 자동으로 모니터링하여 서비스 업데이트가 제공되면 이를 적용합니다. 일반적으로 MemoryDB가 이 업데이트를 적용할 수 있도록 미리 정의된 유지 관리 기간을 설정합니다. 그러나 경우에 따라 이 접근 방식이 너무 엄격하여 비즈니스 흐름이 제한될 수 있습니다.

MemoryDB의 서비스 업데이트를 통해 적용할 업데이트와 적용 시기를 제어할 수 있습니다. 선택한 MemoryDB 클러스터에 대한 업데이트 진행 상황을 실시간으로 모니터링할 수도 있습니다.

서비스 업데이트 관리

MemoryDB 서비스 업데이트는 정기적으로 릴리스됩니다. 서비스 업데이트에 대해 검증된 클러스터가 하나 이상 있는 경우 업데이트가 릴리스되면 이메일, SNS, PHD(Personal Health Dashboard) 및 Amazon CloudWatch Events를 통해 알림을 받습니다. 업데이트는 MemoryDB 콘솔의 서비스 업데이트 페이지에도 표시됩니다. 이 대시보드를 사용하여 모든 서비스 업데이트와 MemoryDB 플릿의 상태를 볼 수 있습니다.

자동 업데이트가 시작되기 전에 업데이트 시작 전에 업데이트 적용 시기를 제어합니다. MemoryDB가 항상 현행 보안 패치로 최신 상태를 유지할 수 있도록 가능한 한 빨리 보안 업데이트(security-update) 유형의 업데이트를 모두 적용할 것을 강력하게 권장합니다.

다음 섹션에서는 다음과 같은 옵션에 대해 상세히 알아봅니다.

주제

- Amazon MemoryDB 관리형 유지 관리 및 서비스 업데이트 개요

Amazon MemoryDB 관리형 유지 관리 및 서비스 업데이트 개요

인스턴스에 원활하게 적용되는 패치 및 업그레이드를 통해 MemoryDB 플릿을 주기적으로 업그레이드합니다. 이 작업은 두 가지 방법으로 수행할 수 있습니다.

1. 지속적인 관리형 유지 관리.
2. 서비스 업데이트.

보안, 신뢰성, 운영 성능을 강화하는 업그레이드 적용에 있어 이러한 유지 관리 및 서비스 업데이트가 필요합니다.

지속적인 관리형 유지 관리는 사용자가 별도의 조치를 취할 필요 없이 유지 관리 기간에 종종 직접 수행됩니다. 유지 관리 기간은 모든 고객에게 필수이며 옵트아웃할 수 있는 옵션이 없다는 점에 유의해야

합니다. 이러한 설정된 유지 관리 기간 동안 중요하거나 중요한 활동을 피하는 것이 좋습니다. 또한 시스템의 보안 및 최적의 성능을 보장하기 위해 중요한 업데이트를 건너뛸 수 없습니다.

서비스 업데이트는 사용자가 직접 적용할 수 있는 유연성을 제공합니다. 기한이 정해져 있으며 유지 관리 기간으로 이동하여 기한이 경과한 후 적용할 수 있습니다.

업데이트는 교체 시 자동으로 적용되므로 가급적 빨리 적용하거나 노드를 교체하여 관리할 수 있습니다. 업데이트가 이전의 모든 노드에 적용된 경우 들어오는 유지 관리 기간 동안 업데이트 활동이 발생하지 않습니다.

서비스 업데이트

MemoryDB의 서비스 업데이트를 사용하면 재량에 따라 특정 서비스 업데이트를 적용할 수 있습니다. 이러한 업데이트에는 다음과 같이 보안 패치 또는 소프트웨어 소규모 업데이트의 유형이 있습니다. 이러한 업데이트는 클러스터의 보안, 신뢰성, 운영 성능을 강화하는 데 도움이 됩니다.

이러한 서비스 업데이트의 가치는 업데이트를 적용할 시기를 제어할 수 있다는 것입니다(예: MemoryDB 클러스터의 연중무휴 가용성이 필요한 중요한 비즈니스 이벤트가 있는 경우 서비스 업데이트 적용을 지연시킬 수 있음).

서비스 업데이트에 대해 검증된 클러스터가 하나 이상 있는 경우 업데이트가 릴리스되면 이메일, Amazon SNS, AWS Health 대시보드, Amazon CloudWatch Events 이벤트를 통해 알림을 받습니다. 업데이트는 MemoryDB 콘솔의 서비스 업데이트 페이지에도 표시됩니다. 이 대시보드를 사용하여 모든 서비스 업데이트와 MemoryDB 플릿의 상태를 볼 수 있습니다.

자동 업데이트가 시작되기 전에 업데이트 시작 전에 업데이트 적용 시기를 제어합니다. MemoryDB가 항상 현행 보안 패치로 최신 상태를 유지할 수 있도록 가능한 한 빨리 보안 업데이트(security-update) 유형의 업데이트를 모두 적용할 것을 강력하게 권장합니다.

클러스터는 다른 서비스 업데이트의 일부일 수 있습니다. 대부분의 업데이트는 별도로 적용할 필요가 없습니다. 클러스터에 하나의 업데이트를 적용하면 해당하는 경우 다른 업데이트가 완료된 것으로 표시됩니다. 상태가 자동으로 “완료됨”으로 변경되지 않으면 동일한 클러스터에 여러 업데이트를 별도로 적용해야 할 수 있습니다.

서비스 업데이트의 영향 및 가동 중지 시간

사용자 또는 Amazon MemoryDB가 하나 이상의 MemoryDB 클러스터에 서비스 업데이트를 적용하면 선택한 모든 클러스터가 업데이트될 때까지 각 샤드 내에서 한 번에 하나 이하의 노드에 업데이트가 적용됩니다. 업데이트 중인 노드에는 몇 초의 가동 중지 시간이 발생하는 반면, 나머지 클러스터는 트래픽을 계속 처리합니다.

- 클러스터 구성에는 변경 사항이 없습니다.
- CloudWatch 지표에 지연이 표시되며 가능한 한 빨리 따라잡게 됩니다.

노드 교체는 애플리케이션에 어떤 영향을 미치나요? - MemoryDB 노드의 경우 교체 프로세스는 내구성과 가용성을 보장하도록 설계되었습니다. 단일 노드 MemoryDB 클러스터의 경우 MemoryDB는 복제본을 동적으로 가동하고 내구성 구성 요소에서 데이터를 복원한 다음 장애 조치합니다. 여러 노드로 구성된 복제 그룹의 경우 MemoryDB는 기존 복제본을 대체하고 내구성 구성 요소의 데이터를 새 복제본과 동기화합니다. MemoryDB는 노드가 2개 이상인 경우에만 다중 AZ이므로 이 시나리오에서는 프라이머리를 교체하면 읽기 전용 복제본으로 장애 조치가 트리거됩니다. 클러스터가 들어오는 쓰기 요청을 처리하는 동안 계획된 노드 교체가 완료됩니다. 노드가 하나뿐인 경우 MemoryDB는 프라이머리 노드를 교체한 다음 내구성 구성 요소의 데이터를 동기화합니다. 이 시간 동안 프라이머리 노드를 사용할 수 없으므로 쓰기 중단이 길어집니다.

원활한 교체 환경을 제공하고 데이터 손실을 최소화하려면 어떤 모범 사례를 따라야 하나요? - MemoryDB에서 데이터는 내구성이 뛰어나며 단일 노드 구현에서도 데이터 손실이 예상되지 않습니다. 그러나 장애가 발생할 경우 손실 가능성을 최소화하기 위해 다중 AZ 및 백업 전략을 구현하는 것이 좋습니다. 원활한 교체를 위해 클러스터를 안정적으로 유지하기 위해 한 번에 동일한 클러스터의 노드만 교체하려고 합니다. 다중 AZ를 활성화하여 여러 가용 영역에서 기본 복제본과 읽기 전용 복제본을 프로비저닝할 수 있습니다. 이 경우 노드가 교체되면 기본 역할이 샤드의 복제본으로 장애 조치됩니다. 이제 이 샤드는 트래픽을 처리하며 데이터는 내구성 구성 요소에서 복원됩니다. 구성에 샤드당 기본 복제본 하나와 단일 복제본 하나가 포함된 경우 패치 적용 전에 복제본을 추가하는 것이 좋습니다. 이렇게 하면 패치 적용 프로세스 중에 가용성이 저하되는 것을 방지할 수 있습니다. 들어오는 쓰기 트래픽이 적은 기간 동안 교체를 예약하는 것이 좋습니다.

유지 관리 중에 애플리케이션 중단을 최소화하려면 어떤 클라이언트 구성 모범 사례를 따라야 합니까? - MemoryDB에서는 클러스터 모드 구성이 항상 활성화되어 관리형 또는 비관리형 작업 중에 최상의 가용성을 제공합니다. 복제본 노드의 개별 노드 엔드포인트는 모든 읽기 작업에 사용할 수 있습니다. MemoryDB에서는 클러스터에서 항상 자동 장애 조치가 활성화되므로 프라이머리 노드가 변경될 수 있습니다. 따라서 애플리케이션에서는 노드의 역할을 확인하고 읽기 엔드포인트를 모두 업데이트하여 프라이머리에서 상당한 로드가 발생하지 않도록 해야 합니다. 마찬가지로 유지 관리 기간 동안 읽기 요청으로 복제본에 과부하를 가하지 마세요. 이를 달성하는 한 가지 방법은 적어도 2개 이상의 읽기 전용 복제본을 마련하여 유지 관리 중 읽기 중단이 일어나지 않도록 하는 것입니다.

클라이언트 애플리케이션을 테스트하여 클라이언트 애플리케이션이 Redis/Valkey 클러스터 프로토콜을 준수하는지 확인하고 요청을 노드 간에 올바르게 리디렉션할 수 있는지 확인하는 것이 중요합니다. 유지 관리 및 교체 활동 중에 MemoryDB 노드에 과부하가 걸리지 않도록 백오프 및 재시도 전략을 구현하는 것이 좋습니다.

일정 변경 - 유지 관리 기간을 변경하여 서비스 업데이트를 연기할 수 있습니다. 예약된 업데이트는 예약된 날짜가 클러스터의 유지 관리 기간과 일치하는 경우에만 클러스터에 적용됩니다. 유지 관리 기간을 변경하고 예약된 날짜가 지나면 서비스 업데이트가 다음 주 내에 새로 지정된 기간으로 다시 예약됩니다. 새 날짜에 도달하기 1주일 전에 새 알림을 받게 됩니다.

AWS의 보안은 공동 책임입니다. 최대한 빨리 업데이트를 적용하는 것이 좋습니다.

서비스 업데이트 옵트아웃 - "시작 날짜 자동 업데이트" 속성의 값을 확인하여 서비스 업데이트를 옵트아웃할 수 있는지 확인할 수 있습니다. 서비스 업데이트의 "시작 날짜 자동 업데이트" 속성 값이 설정된 경우 MemoryDB는 예정된 유지 관리 기간에 남은 클러스터로 서비스 업데이트를 예약하므로 옵트아웃할 수 없습니다. 여전히 유지 관리 기간 전에 나머지 클러스터에 서비스 업데이트를 적용하면 MemoryDB는 유지 관리 기간 동안 서비스 업데이트를 다시 적용하지 않습니다. 자세한 내용은 서비스 업데이트 적용 섹션을 참조하세요.

유지 관리 기간 동안 MemoryDB에서 서비스 업데이트를 직접 적용할 수 없는 이유는 무엇입니까? - 서비스 업데이트의 목적은 업데이트 적용 시기에 대한 유연성을 제공하는 것입니다. MemoryDB 지원 규정 준수 프로그램에 참여하지 않는 클러스터는 이러한 업데이트를 적용하지 않거나 연중 축소된 빈도로 적용하도록 선택할 수 있습니다. 그러나 규정을 준수하기 위해 업데이트를 적용하는 것이 좋습니다. 이는 서비스 업데이트의 "시작 날짜 자동 업데이트" 속성 값이 없는 경우에만 해당됩니다. 자세한 내용은 MemoryDB의 규정 준수 확인 섹션을 참조하세요.

유지 관리 기간의 업데이트는 서비스 업데이트와 어떻게 다릅니까? - 지속적인 관리형 유지 관리를 통해 적용된 업데이트는 사용자 측에서 필요한 조치 없이 유지 관리 기간에 직접 예약됩니다. 서비스 업데이트는 시간이 정해져 있으며 "시작 날짜 자동 업데이트"까지 적용할 시기를 제어할 수 있습니다. 해당 시점까지 여전히 적용되지 않는 경우 MemoryDB는 유지 관리 기간에 이러한 업데이트를 예약할 수 있습니다.

지속적인 관리형 유지 관리 업데이트

이러한 업데이트는 필수이며 사용자 측에서 필요한 조치 없이 유지 관리 기간에 직접 적용됩니다. 이러한 업데이트는 서비스 업데이트에서 제공하는 업데이트와는 별개입니다.

지속적인 유지 관리 영향 및 가동 중지 시간

노드 교체에는 얼마나 걸리나요? - 교체는 일반적으로 30분 이내에 완료됩니다. 특정 인스턴스 구성 및 트래픽 패턴에서는 교체가 더 오래 걸릴 수 있습니다.

노드 교체는 애플리케이션에 어떤 영향을 미치나요? - 지속적인 관리형 유지 관리 업데이트는 노드 교체를 통해 "서비스 업데이트"와 동일한 방식으로 적용됩니다. 자세한 내용은 위의 서비스 업데이트 영향 및 가동 중지 시간 섹션을 참조하세요.

노드 교체를 직접 관리하려면 어떻게 해야 합니까? 예정된 노드 교체 주기 이전에 언제든 이러한 교체를 직접 관리할 수 있는 옵션이 있습니다. 교체를 직접 관리하도록 선택한 경우 사용 사례에 따라 다양한 작업을 수행할 수 있습니다.

- 클러스터의 노드를 하나 이상의 샤드로 대체: 백업 및 복원 또는 스케일 아웃을 사용한 다음 스케일 인을 사용하여 노드를 대체할 수 있습니다.
- 유지 관리 기간 변경: 클러스터의 유지 관리 기간도 변경할 수 있습니다. 유지 관리 기간을 이후의 더 편리한 시간으로 변경하려면 UpdateCluster API, update-cluster CLI를 사용하거나 MemoryDB 관리 콘솔에서 수정을 클릭합니다. 유지 관리 기간을 변경하면 MemoryDB는 새로 지정된 기간 동안 유지 관리를 위해 노드를 예약합니다.

  실제 작동 방법을 볼 수 있도록, 지금 11월 9일 목요일 오후 3시고 다음 유지 관리 기간은 11월 10일 금요일 오후 5시라고 가정해 보겠습니다. 다음은 3가지 시나리오입니다.
  - 유지 관리 기간을 금요일 오후 4시로 변경합니다(현재 날짜/시간 이후 및 예약된 다음 유지 관리 기간 이전). 그러면 이 노드는 11월 10일 금요일 오후 4시에 교체됩니다.
  - 유지 관리 기간을 토요일 오후 4시로 변경합니다(현재 날짜/시간 이후 및 예약된 다음 유지 관리 기간 이후). 그러면 이 노드는 11월 11일 토요일 오후 4시에 교체됩니다.
  - 유지 관리 기간을 수요일 오후 4시로 변경합니다(현재 날짜/시간보다 일주일 빠름). 그러면 이 노드는 다음주 수요일인 11월 15일 오후 4시에 교체됩니다.

  자세한 내용은 유지 관리 관리 중 섹션을 참조하세요.

  서로 다른 리전의 서로 다른 클러스터에 있는 노드를 동시에 교체할 수 있습니다. 단, 이러한 클러스터의 유지 관리 기간이 동일하게 구성되어 있어야 합니다.

예정된 교체 일정을 확인하려면 어떻게 해야 하나요? - 상태 대시보드에서 AWS 상태 알림을 받아야 합니다. 또한 DescribeServiceUpdates API를 사용하여 다양한 서비스 업그레이드의 상태를 확인할 수 있습니다. 예측 가능한 교체에 대해 고객에게 사전에 알리기 위해 모든 노력을 기울이고 있습니다. 그러나 예측할 수 없는 장애와 같은 예외적인 경우에는 예고 없는 교체가 있을 수 있습니다.

더 적합한 시간에 예약된 유지 관리를 변경할 수 있습니까? - 예, 유지 관리 기간을 변경하여 예약된 유지 관리를 더 적절한 시간으로 연기할 수 있습니다.

이러한 노드 교체를 수행하는 이유는 무엇입니까? - 이러한 교체는 기본 호스트에 필수 소프트웨어 업데이트를 적용하는 데 필요합니다. 업데이트는 보안, 신뢰성, 운영 성능을 강화하는 데 도움이 됩니다.

이러한 교체가 여러 가용 영역의 노드와 여러 리전의 클러스터에 동시에 영향을 미칩니까? - 클러스터의 유지 관리 기간에 따라 여러 가용 영역 또는 리전에서 병렬로 교체를 실행할 수 있습니다.

서비스 업데이트 적용

업데이트가 사용 가능(Available) 상태일 때부터 플릿에 서비스 업데이트를 적용할 수 있습니다. 서비스 업데이트는 축적됩니다. 즉, 아직 적용되지 않은 모든 업데이트가 최신 업데이트에 포함됩니다.

서비스 업데이트에 자동 업데이트가 사용 설정되어 있는 경우 사용 가능하게 되면 아무런 조치도 취하지 않도록 선택할 수 있습니다. MemoryDB는 Auto-update start date(자동 업데이트 시작일) 이후에 클러스터의 유지 관리 기간 동안 업데이트를 적용하도록 예약합니다. 업데이트의 각 단계에 대한 관련 알림을 받게 됩니다.

> **Note**
>
> 사용 가능(Available) 또는 예약됨(Scheduled) 상태인 서비스 업데이트만 적용할 수 있습니다.

적용 가능한 MemoryDB 클러스터에 서비스별 업데이트의 검토 및 적용에 관한 자세한 정보는 [콘솔을 사용한 서비스 업데이트 적용](#) 섹션을 참조하세요.

하나 이상의 MemoryDB 클러스터에 새로운 서비스 업데이트가 제공될 경우, MemoryDB 콘솔, API 또는 AWS CLI을(를) 사용하여 이 업데이트를 적용할 수 있습니다. 다음 섹션에서는 업데이트 적용에 사용할 수 있는 옵션에 대해 설명합니다.

콘솔을 사용한 서비스 업데이트 적용

다른 정보와 함께 사용 가능한 서비스 업데이트 목록을 보려면 콘솔의 서비스 업데이트 페이지로 이동하세요.

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/memorydb/에서 MemoryDB 콘솔을 엽니다.
2. 탐색 창에서 서비스 업데이트(Service Updates)를 선택합니다.

서비스 업데이트(Service Updates)에서 다음 내용을 볼 수 있습니다.

- 서비스 업데이트 이름(Service update name): 서비스 업데이트의 고유 이름입니다.
- 업데이트 설명: 서비스 업데이트에 관한 상세 정보를 제공합니다.
- 자동 업데이트 시작일: 이 속성을 설정하면 MemoryDB는 이 날짜 이후 적절한 유지 관리 기간에 클러스터가 자동 업데이트되도록 스케줄링하기 시작합니다. 정확한 예정된 유지 관리 기간에 대한 알림을 미리 받게 되며, 이 기간은 자동 업데이트 시작일 직후가 아닐 수도 있습니다. 원하는 경우, 언제

든지 클러스터에 업데이트를 적용할 수 있습니다. 속성이 설정되지 않은 경우, 서비스 업데이트는 자동 업데이트가 활성화되지 않으며 MemoryDB는 클러스터를 자동으로 업데이트하지 않습니다.

클러스터 업데이트 상태(Cluster update status) 섹션에서 서비스 업데이트가 적용되지 않았거나 최근에 막 적용된 클러스터 목록을 볼 수 있습니다. 각 클러스터에 대해 다음을 볼 수 있습니다.

- 클러스터 이름(Cluster name) - 클러스터의 이름입니다.
- 업데이트된 노드(Nodes updated): 업데이트되었거나 특정 서비스 업데이트를 여전히 사용할 수 있는 특정 클러스터 내 개별 노드의 비율입니다.
- 업데이트 유형(Update Type): 보안 업데이트(security-update) 또는 엔진 업데이트(engine-update) 중 하나에 해당하는 서비스 업데이트 유형
- 상태(Status): 클러스터의 서비스 업데이트의 상태로, 다음 중 하나에 해당합니다.
  - 사용 가능: 업데이트를 필요한 클러스터에 사용할 수 있습니다.
  - 진행 중: 업데이트가 이 클러스터에 적용 중입니다.
  - 예약됨: 업데이트 날짜가 예약되었습니다.
  - 완료: 업데이트가 성공적으로 적용되었습니다. 완료 상태의 클러스터는 완료 후 7일 동안 표시됩니다.

  사용 가능(Available) 또는 예약됨(Scheduled) 상태의 클러스터 중 일부 또는 전부를 선택한 다음 지금 적용(Apply now)을 선택하면 해당 클러스터에 업데이트가 적용되기 시작합니다.

AWS CLI를 사용하여 서비스 업데이트 적용

서비스 업데이트가 제공된다는 알림을 받은 후 AWS CLI를 사용하여 해당 업데이트를 검사하고 적용할 수 있습니다.

- 사용 가능한 서비스 업데이트에 대한 설명을 검색하려면 다음 명령을 실행합니다.

  ```
  aws memorydb describe-service-updates --status available
  ```

  자세한 정보는 describe-service-updates를 참조하세요.
- 클러스터 목록에 서비스 업데이트를 적용하려면 다음 명령을 실행합니다.

  ```
  aws memorydb batch-update-cluster --service-update
  ServiceUpdateNameToApply=sample-service-update --cluster-names cluster-1
  cluster2
  ```

자세한 정보는 batch-update-cluster를 참조하세요.

레퍼런스

이 단원의 주제에서는 MemoryDB API 및 AWS CLI의 MemoryDB 섹션 관련 작업을 다룹니다. 또한 여기에는 일반적인 오류 메시지와 서비스 알림에 관한 설명이 포함되어 있습니다.

- MemoryDB API 사용
- MemoryDB API Reference
- AWS CLI Reference의 MemoryDB 섹션

MemoryDB API 사용

이 단원에서는 MemoryDB 작업 사용 방법 및 구현에 관해 작업 중심으로 설명합니다. 이러한 작업에 대한 자세한 설명은 MemoryDB API 참조를 참조하세요.

주제

- Query API 사용
- 사용 가능한 라이브러리
- 애플리케이션 문제 해결

Query API 사용

쿼리 파라미터

HTTP 쿼리 기반 요청은 GET 또는 POST와 같은 HTTP 동사와 `Action` 쿼리 매개 변수를 사용하는 HTTP 요청입니다.

각 쿼리 요청은 인증 및 작업을 처리할 수 있도록 일부 공통 파라미터를 포함해야 합니다.

일부 작업은 파라미터의 목록을 허용합니다. 이러한 목록은 `param.n` 표기법을 사용하여 지정됩니다. *n*의 값은 1부터 시작하는 정수입니다.

쿼리 요청 인증

HTTPS를 통해서만 쿼리 요청을 보낼 수 있으며 모든 쿼리 요청에는 서명이 포함되어야 합니다. 이 섹션에서는 서명을 작성하는 방법을 설명합니다. 아래 절차에 설명된 방법은 서명 버전 4라고 합니다.

다음은 AWS에 대한 요청을 인증하는 데 사용되는 기본 단계입니다. 이 경우, 사용자가 AWS에 등록되어 있으며 액세스 키 ID 및 비밀 액세스 키를 가지고 있다고 가정합니다.

쿼리 인증 절차

1. 발신자가 AWS에 대한 요청을 구성합니다.
2. 발신자가 이 항목의 다음 섹션에 정의된 방법으로 SHA-1 해시 기능을 사용하는 HMAC(Hash-based Message Authentication Code)에 대한 키 해싱인 요청 서명을 계산합니다.
3. 요청의 발신자가 요청 데이터, 서명 및 액세스 키 ID(사용된 비밀 액세스 키의 키 식별자)를 AWS 으(로) 보냅니다.

4. AWS는 액세스 키 ID를 사용하여 보안 액세스 키를 찾습니다.
5. AWS는 요청의 서명 계산에 사용된 동일한 알고리즘을 사용하여 요청 데이터 및 비밀 액세스 키에서 서명을 생성합니다.
6. 서명이 일치하는 경우, 요청이 인증되는 것으로 간주됩니다. 서명이 일치하지 않을 경우, 요청이 삭제되고 AWS에서 오류 응답을 반환합니다.

> **Note**
>
> `Timestamp` 매개 변수가 요청에 포함된 경우, 요청에 대해 계산된 서명은 그 매개 변수 값보다 15분 후에 만료됩니다.
> `Expires` 매개 변수가 요청에 포함된 경우, 그 서명은 `Expires` 매개 변수에 의해 지정된 시간에 만료됩니다.

요청 서명을 계산하려면

1. 정규화된 쿼리 문자열을 만듭니다. 이 절차의 뒷부분에서 필요합니다.
   a. UTF-8 쿼리 문자열 구성 요소를 매개 변수 이름의 일반 바이트 순서로 정렬합니다. 이 매개 변수는 GET URI 또는 POST 요청 본문(Content-Type이 application/x-www-form-urlencoded일 경우)의 내용이 사용될 수 있습니다.
   b. 다음 규칙에 따라 매개 변수 이름과 값을 URL 인코딩합니다.
      i. RFC 3986에 정의된 예약되지 않은 모든 문자는 URL 인코딩하지 않습니다. 이러한 예약되지 않은 문자는 A~Z, a~z, 0~9, 하이픈(-), 밑줄(_), 마침표(.) 및 물결표(~)입니다.
      ii. %XY와 같이 모든 기타 문자를 퍼센트 인코딩합니다(여기서 X 및 Y는 16진 문자 0~9 및 대문자 A~F).
      iii. 확장된 UTF-8 문자는 %XY%ZA... 형식으로 퍼센트 인코딩합니다.
      iv. 공백 문자는 %20(일반 인코딩 구조인 +가 아님)으로 퍼센트 인코딩합니다.
   c. 매개 변수 값이 비어있는 경우에도 인코딩된 매개 변수 이름을 인코딩된 매개 변수 값과 등호(=)(ASCII 문자 61)로 구별합니다.
   d. 앰퍼샌드(&)(ASCII 코드 38)로 이름-값 쌍을 구별합니다.
2. 다음의 의사(pseudo) 문법("\n"은 ASCII 줄 바꿈을 나타냄)에 따라 서명할 문자열을 만듭니다.

```
StringToSign = HTTPVerb + "\n" +
ValueOfHostHeaderInLowercase + "\n" +
```

```
HTTPRequestURI + "\n" +
CanonicalizedQueryString <from the preceding step>
```

HTTPRequestURI 구성 요소는 URI의 HTTP 절대 경로 구성 요소이고 쿼리 문자열은 포함하지는 않습니다. HTTPRequestURI가 비어있는 경우, 슬래시(/)를 사용합니다.

3. 사용자의 보안 액세스 키를 키로, SHA256 또는 SHA1을 해시 알고리즘으로 하여 방금 만든 문자열로 RFC 2104 호환 HMAC를 계산합니다.

   자세한 내용은 https://www.ietf.org/rfc/rfc2104.txt를 참조하세요.
4. 결과 값을 base64로 변환합니다.
5. 요청에서 `Signature` 매개 변수 값을 값으로 포함합니다.

예를 들어, 다음은 샘플 요청입니다(줄 바꿈이 명확성을 위해 추가됨).

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DescribeClusters
    &ClusterName=myCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2021-01-01
```

이전 쿼리 문자열의 경우, 다음 문자열을 통해 HMAC 서명을 계산합니다.

```
GET\n
    memory-db.amazonaws.com\n
    Action=DescribeClusters
    &ClusterName=myCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE%2F20140523%2Fus-east-1%2Fmemorydb%2Faws4_request
    &X-Amz-Date=20210801T223649Z
    &X-Amz-SignedHeaders=content-type%3Bhost%3Buser-agent%3Bx-amz-content-sha256%3Bx-
amz-date
        content-type:
        host:memory-db.us-east-1.amazonaws.com
        user-agent:ServicesAPICommand_Client
```

```
    x-amz-content-sha256:
    x-amz-date:
```

이 결과는 다음의 서명된 요청입니다.

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DescribeClusters
    &ClusterName=myCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20141201/us-east-1/memorydb/aws4_request
    &X-Amz-Date=20210801T223649Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=2877960fced9040b41b4feaca835fd5cfeb9264f768e6a0236c9143f915ffa56
```

서명 프로세스 및 요청 서명 계산에 대한 자세한 내용은 서명 버전 4 서명 프로세스 항목과 그 하위 항목을 참조하세요.

사용 가능한 라이브러리

AWS는 Query API 대신 언어별 API를 사용하여 애플리케이션을 구축하려는 소프트웨어 개발자를 위한 소프트웨어 개발 키트(SDK)를 제공합니다. 이러한 SDK는 보다 쉽게 시작하도록 요청 인증, 요청 재시도 및 오류 처리 같은 기본 기능(API에는 포함되지 않음)을 제공합니다. SDK 및 추가 리소스는 다음 프로그래밍 언어에 대해 제공됩니다.

- Java
- Windows 및 .NET
- PHP
- Python
- Ruby

다른 언어에 대한 자세한 내용은 샘플 코드 및 라이브러리를 참조하세요.

애플리케이션 문제 해결

MemoryDB는 MemoryDB API와 상호 작용하는 동안 발생하는 문제를 해결할 때 도움이 되도록 구체적이고 서술적인 오류를 제공합니다.

오류 검색

일반적으로 사용자는 시간을 소비하여 결과를 처리하기 전에 애플리케이션이 먼저 해당 요청으로 오류가 발생되는지 여부를 확인하려고 합니다. 오류 발생 여부를 확인하는 가장 쉬운 방법은 MemoryDB API의 응답에서 `Error` 노드를 찾는 것입니다.

XPath 구문은 `Error` 노드의 발생뿐만 아니라 오류 코드 및 메시지를 쉽게 검색할 수 있는 간단한 방법을 제공합니다. 다음 코드 조각에서는 요청 중에 오류가 발생했는지 여부를 파악하기 위해 Perl 및 XML::Xpath 모듈을 사용합니다. 오류가 발생되면 코드는 응답에 첫 번째 오류 코드와 메시지를 인쇄합니다.

```
use XML::XPath;
my $xp = XML::XPath->new(xml =>$response);
if ( $xp->find("//Error") )
{print "There was an error processing your request:\n", " Error code: ",
$xp->findvalue("//Error[1]/Code"), "\n", " ",
$xp->findvalue("//Error[1]/Message"), "\n\n"; }
```

문제 해결 팁

다음 절차를 통해 MemoryDB API의 문제를 진단하고 해결하는 것이 좋습니다.

- MemoryDB가 올바르게 실행되는지 확인합니다.

  이렇게 하려면, 브라우저 창을 열고 MemoryDB 서비스(https://memory-db.us-east-1.amazonaws.com 등)에 쿼리 요청을 제출하면 됩니다. MissingAuthenticationTokenException 또는 UnknownOperationException은 서비스가 사용 가능하고 요청에 응답하는지 확인할 수 있습니다.
- 요청 구조 확인.

  각 MemoryDB 작업에 대한 참조 페이지는 MemoryDB API 참조에 있습니다. 파라미터를 올바르게 사용하고 있는지 여부를 다시 확인합니다. 어떤 문제가 발생할 수 있을 지에 대해 미리 알아보려면 샘플 요청이나 사용자 시나리오를 살펴보고 이러한 샘플이 유사한 작업을 하고 있는지 확인하세요.
- 포럼 확인.

MemoryDB에는 그 과정에서 다른 사람들이 경험한 문제에 대한 해결책을 검색할 수 있는 토론 포럼이 있습니다. 포럼을 보려면 다음 사이트를 참조하세요.

https://forums.aws.amazon.com/.

MemoryDB 할당량

AWS 계정에는 각 AWS 서비스에 대한 기본 할당량(이전에는 제한이라고 함)이 있습니다. 다르게 표시되지 않는 한, 리전별로 각 할당량이 적용됩니다. 일부 할당량에 대한 증가를 요청할 수 있으며 다른 할당량은 늘릴 수 없습니다.

할당량 증가를 요청하려면 Service Quotas 사용 설명서의 [할당량 증가 요청](#)을 참조하십시오. Service Quotas에서 아직 할당량을 사용할 수 없는 경우 [한도 증가 양식](#)을 사용합니다.

AWS 계정에는 MemoryDB와 관련하여 다음과 같은 할당량이 있습니다.

이름	기본값	설명	지표 이름
리전당 노드	300	한 리전의 모든 MemoryDB 클러스터에 있는 노드의 최대 수 이 할당량은 지정된 리전 내에서 예약된 노드와 예약되지 않은 노드에 적용됩니다. 동일한 노드에 최대 300개의 예약 노드 및 300개의 예약되지 않은 노드를 포함할 수 있습니다.	NodesPerRegion
클러스터당 노드 수 (Redis OSS 클러스터 모드 활성화됨)	90	MemoryDB용 개별 Redis OSS 클러스터에 있는 노드의 최대 수	NodesPerCluster
리전당 파라미터 그룹	300	한 리전에서 생성할 수 있는 최대 파라미터 그룹 수입니다.	ParameterGroup

이름	기본값	설명	지표 이름
리전당 서브넷 그룹	300	한 리전에서 생성할 수 있는 최대 서브넷 그룹 수입니다.	SubnetGroup
서브넷 그룹당 서브넷 수	20	한 서브넷 그룹에 대해 정의할 수 있는 최대 서브넷 수입니다.	SubnetsPerSubnetGroup
리전당 사용자	2000	한 리전에서 생성할 수 있는 최대 사용자 수입니다.	User
리전당 사용자 그룹	200	한 리전에서 생성할 수 있는 최대 사용자 그룹 수입니다.	UserGroup
사용자 그룹당 사용자	100	한 사용자 그룹에 대해 정의할 수 있는 최대 사용자 수입니다.	UsersPerUserGroup

MemoryDB 사용 설명서에 대한 문서 기록

다음 표에서는 MemoryDB에 대한 문서 릴리스를 소개합니다.

변경 사항	설명	날짜
MemoryDB Multi-Region이 시작되었습니다	MemoryDB Multi-Region이 시작되었습니다.	2024년 12월 1일
MemoryDB 다중 리전에 대한 IAM 및 보안 정책 업데이트	IAM 및 보안 정책이 업데이트되었습니다. 자세한 내용은 서비스 연결 역할 사용 및 서비스 연결 역할 사용 섹션을 참조하십시오.	2024년 12월 1일
이제 MemoryDB가 Valkey를 지원합니다	이제 MemoryDB가 Valkey를 지원합니다.	2024년 10월 8일
이제 MemoryDB가 IAM을 사용한 사용자 인증을 지원합니다	IAM 인증을 통해 AWS Identity and Access Management ID를 사용하여 MemoryDB 연결을 인증할 수 있습니다. 그러면 보안 모델이 강화되고 여러 보안 관리 작업이 단순화됩니다. 자세한 내용은 IAM을 통한 인증을 참조하세요.	2023년 5월 10일
이제 MemoryDB가 Redis OSS 7을 지원합니다.	이번 릴리스로 Redis OSS 함수, ACL 개선 사항, 샤딩된 Pub/Sub, 향상된 I/O 멀티플렉싱 등 여러 새로운 기능이 MemoryDB에 도입되었습니다. 자세한 정보는 Redis OSS 엔진 버전을 참조하세요.	2023년 5월 9일
이제 MemoryDB는 예약 노드를 제공합니다	예약 노드는 온디맨드 노드 요금과 비교하여 대폭 할인된 요금을 제공합니다. 예약 노드	2022년 12월 27일

	는 물리적 노드가 아니며 계정에서 온디맨드 노드를 사용할 때 적용되는 결제 할인에 가깝습니다. 자세한 내용은 MemoryDB 예약 노드 섹션을 참조하세요.	
MemoryDB는 이제 데이터 계층화를 지원합니다	MemoryDB 데이터 계층화. 저렴한 방법으로 데이터 계층화를 사용하여 클러스터를 최대 수백 테라바이트 용량으로 확장할 수 있습니다. 자세한 내용은 데이터 계층화를 참조하세요.	2022년 11월 3일
이제 MemoryDB가 기본 JavaScript Object Notation(JSON) 형식을 지원합니다	기본 JSON(JavaScript Object Notation) 형식은 Redis OSS 클러스터 내에서 복잡한 데이터세트를 인코딩하는 간단한 스키마리스 방법입니다. Redis OSS 클러스터 내에서 JSON(JavaScript Object Notation) 형식을 사용하여 데이터를 저장하고 액세스하고, 직렬화 및 역직렬화를 위해 사용자 지정 코드를 관리할 필요 없이 해당 클러스터에 저장된 JSON 데이터를 업데이트할 수 있습니다. 자세한 정보는 JSON 시작하기를 참조하세요.	2022년 5월 25일

MemoryDB는 이제 AWS PrivateLink를 지원합니다	AWS PrivateLink를 사용하면 인터넷 게이트웨이, NAT 디바이스, VPN 연결 또는 AWS Direct Connect 연결 없이 MemoryDB API 작업에 비공개로 액세스할 수 있습니다. 자세한 내용은 MemoryDB 및 인터페이스 VPC 엔드포인트(AWS PrivateLink)를 참조하세요.	2022년 1월 24일
최초 릴리스입니다.	MemoryDB 사용 설명서의 최초 릴리스입니다. 자세한 내용은 MemoryDB란 무엇입니까?를 참조하세요.	2021년 8월 19일

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